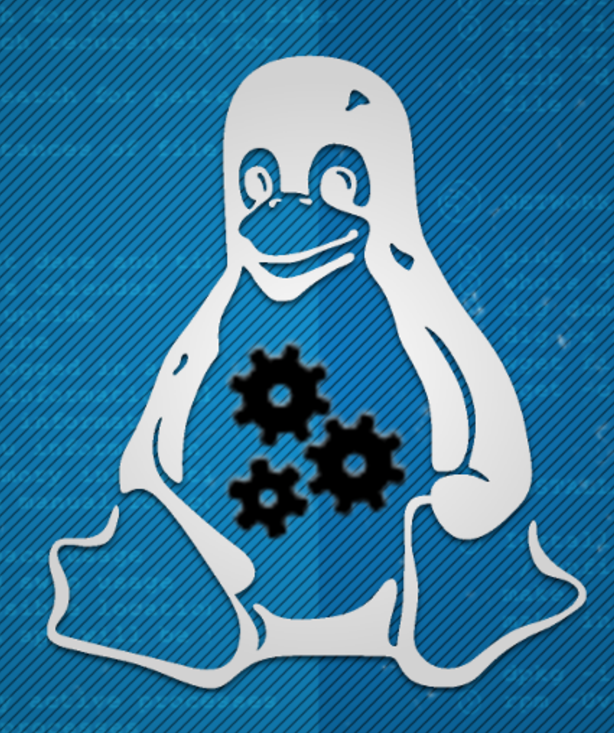




الإدارة المتقدمة لجنو/لينكس



تأليف : "د. رمو سـپي بلُدريتو" و "د. جسـپ جبُرا إستيفَ"

ترجمة : "عبدالرحيم غالب الفاخوري"

GNU FDL

الإدارة المتقدمة لجنو/لينكس

د. رمو ســـپي بُلدريتو (مؤلف)   د. جســـــپ جُبرا إستيڤَ (مؤلف)

عبدالرحيم غالب فاخوري (مترجِم)

٢٠١٣

د. جســپ جُبرا إستيڤَ (مؤلف)

- مهندس خبير و دكتوراه في تقنية المعلومات من UAB.

- مدرس تقنية المعلومات، والدراسات المتعلقة بالاتصالات، والوسائط المتعددة في الجامعة المفتوحة لكَالونيا UOC.

د. رمو سـپي بُلدريتو (مؤلف)

- مهندس اتصالات

- دكتوراه في تقنية المعلومات من UAB.

- مدرس دائرة معمارية الحاسوب وأنظمة التشغيل في UAB.

عبدالرحيم غالب فاخوري (مترجِم)

بكالوريوس في اللغة الإنجليزية، فرعي اللغة الفرنسية - جامعة الخليل

عضو ومشرف في مجتمع لينكس العربي.

عضو فريق عربايز، ومنسق تدريب المساهمين الجدد.

منسق توطين اللغة العربية في مشروع بيئة سطح المكتب KDE.

إخلاء مسؤولية:

بذل القائمون على هذا العمل أقصى جهودهم لتحقيق مستوى عالٍ من الجودة، إلا أنهم لا يتحملون أية مسؤولية ولا يوفرون أيّة ضمانات صريحة أو ضمنية تجاه ما قد ينجم عن استخدام أو سوء استخدام ما ورد في هذا الكتاب.

جميع الحقوق محفوظة © 2013، عبدالرحيم غالب فاخوري و مجتمع لينكس العربيّ
يسمح لك بنسخ أو توزيع أو تعديل هذا المستند
وفق شروط اتفاقية رخصة غنو للمستندات الحرة GNU FDL الاصدار 1.2
أو أي اصدارٍ لاحقٍ تم نشره من قبل مؤسسة البرمجيات الحرة،
دون أية أقسام ثابتة، نصوص غلاف أمامي ونصوص غلاف خلفي.
لقد تمت إضافة نسخة من اتفاقية الرخصة في القسم
المعنون "رخصة غنو للمستندات الحرة GNU FDL".

تعود ملكية تصميم الغلاف للعضو saf1 في مجتمع لينكس العربي.
كل العلامات التجارية المذكورة تعود لأصحابها.

Copyright © 2013, Abdalrahim Fakhouri & Linux Arabs Community.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled "GNU
Free Documentation License".

The design of the cover belongs to the member “saf1” in Linux Arabs Community.
All trademarks and copyrights refer to their respective owners.

شكر وعرفان

أحمد الله على توفيقه في إنجاح هذا العمل، راجياً أن يكون خالصاً لوجهه الكريم. وأتقدم بجزيل الشكر لوالديّ على دعمهما المتواصل في مسيرتي التعليمية والحياتية، ولأكاديمية التقنية الحرّة Free Technology Academy على الكتّاب الذي يعدّ بحق أحد أفضل المراجع المتعلقة بأنظمة جنو/لينكس عالمياً، ولمجتمع لينُكس العربي إدارة ومشرفين وأعضاء، على دعمهم المتواصل للخروج بالنسخة العربية من هذا الكتّاب. وأشكر أيضاً مجتمع البرمجيات الحرّة ومفتوحة المصدر لتوفيرهم الأدوات التي اعتمدت عليها في تحرير هذا الكتّاب. وأتقدم بجزيل الشكر أيضاً لكل من ساهم في دعم هذا الكتّاب مادياً أو معنوياً.

والله الموفق!

المترجِم

عبدالرحيم غالب الفاخوري

السبت ٢٣/٣/٢٠١٣

١٠ جمادى الأولى ١٤٣٤

تمهيد

لقد أصبحت البرمجيات مورداً اجتماعيًّا استراتيجيًّا في العقود القليلة الأخيرة. إن الضرورة الملحّة للبرمجيات الحرّة – التي دخلت في القطاعات رئيسيّة من سوق تقنية المعلومات والاتصالات ICT – أخذت تغير الاقتصاد المتعلق بتطوير واستخدام البرمجيات بشكل كبير. إن البرمجيات الحرّة Free Software – والتي يشار إليها أيضاً بالمصدر المفتوح Open Source وأحياناً أخرى بعبارة Libre Software – يمكن استخدامها، ودراستها، ونسخها، وتعديلها، ونشرها بحرّية. وتوفّر البرمجيات الحرّة حريّة التعلم والتعليم دون الانخراط في الاعتماد على أيّ موفر وحيد للتقنية. تعدّ هذه الحريات متطلبات مسبقة أساسيّة للتنمية المستدامة والمجتمع المعلوماتي الشامل.

رغم وجود إقبال متزايد على التقنيات الحرّة (البرمجيات الحرة والمعايير المفتوحة)، فما تزال نسبة من لديهم معرفة وخبرة كافية في هذا المجال محدودة. ولهذا يحاول كل من أكاديمية التقنية الحرّة Free Technology Academy (اختصاراً FTA) ومجتمع لينكس العربيّ الاستجابة لهذه الحاجة.

تعريف بأكاديمية التقنية الحرة

إن أكاديمية التقنية الحرة FTA هي مبادرة مشتركة من عدد من المؤسسات التعليمية في دول عديدة. وهي تحاول المساهمة في مجتمع يسمح لكل مستخدميه بالدراسة، والمشاركة، والبناء على المعرفة الموجودة دون قيود.

تعريف بمجتمع لينكس العربيّ

إن مجتمع لينكس العربي هو مجتمع تعاونيّ حرّ يُعنى بدعم البرمجيات الحرّة ومفتوحة المصدر وأنظمة جنو/لينكس في العالم العربي وتوفير توثيق وتقديم دعم فني لها. ويوفر منتدى متخصصاً، وموسوعة، وصفحات على الشبكات الاجتماعيّة ومواقع التدوين المصغّر، وقناة دعم فني حيّ على شبكة المحادثة الفورية IRC (القناة #linuxac على freenode.net). الموقع الرسميّ لمجتمع لينك العربيّ هو http://www.linuxac.org.

ماذا تقدم FTA ؟

توفر الأكاديمية برنامجاً بمستوى يكافئ الدراسات العليا بوحدات مساقات عن التقنيات الحرّة. يمكن للدارسين أن يختاروا دخول مساق منفرد، أو التسجيل للبرنامج بأكمله. ويتم التدريس عبر الإنترنت في الحرم الجامعي الافتراضي للأكاديمية، والمكون من طاقم تدريس من الجامعات المشاركة. الساعات المعتمدة التي يتم الحصول عليها في برنامج الأكاديمية معترف بها في هذه الجامعات.

من القائمون على الأكاديمية؟

لقد تأسست الأكاديمية عام 2008 بدعم من برنامج التعليم مدى الحياة LLP للاتحاد الأوروبيّ، ضمن شرط كونها Free Knowledge Institute، وبالتعاون مع الجامعات الأوروبية: Open Universiteit Nederland (هولندا)، و Universitat Oberta de Catalunya (إسبانيا)، و University of Agder (النرويج).

لمن الأكاديمية؟

إن أكاديمية التقنية الحرة موجهة خصيصاً لمن لهم علاقة بمجال تقنية المعلومات من المتخصصين، والمدرسين، والطلاب، وصانعي القرار.

ماذا عن التراخيص؟

كل المواد التعليمية التي تستخدمها أو تطورها الاكاديمية هي مصادر تعليمية مفتوحة Open Educational Resources، ومنشورة ضمن تراخيص حرة copyleft تسمح باستخدامها وتعديلها ونشرها بحرّيّة. وكذلك، فالبرمجيات المستخدمة في الحرم الجامعي الافتراضي للأكاديمية هي برمجيات حرة مبنية ضمن إطار عمل المعايير المفتوحة.

تطور هذا الكتاب

لقد أعادت الأكاديمية استخدام مواد تعليمية موجودة من جامعة Universitat Oberta de Catalunya والتي تم تطويرها بمشاركة طاقم LibreSoft من جامعة Universidad Rey Juan Carlos. في 2008، تمت ترجمة هذا الكتاب إلى الإنجليزيّة بمساعدة مشروع "العلوم والتدريس والتعليم بحرّيّة" SELF، وبدعم من برنامج إطار العمل السادس للاتحاد الأوروبيّ. وعام 2009، قامت أكاديمية التقنية الحرّة بتحسين هذا الكتاب. إضافة إلى هذا، فقد طوت الأكاديمية دليلاً دراسيًّا وأنشطة تعليميّة متاحة للدارسين الملتحقين بالأكاديمية.

أما عن النسخة العربية، فهي إحدى جوانب العمل الدؤوب لمجتمع لينكس العربي في تقديم كل ما فيه فائدة من تطبيقات وترجمات ومصادر تعليمية ودعم فنيّ للبرمجيات الحرّة ومفتوحة المصدر وأنظمة جنو/لينكس وأشباه يونكس. ففي ظل محدودية الموارد التعليمية العربية، وافتقارها إلى المحتوى التقنيّ المتخصص، ولعدم توفر كتاب مكافئ باللغة العربية، وبمساندة المساهمين (جزاهم الله خيراً)، تمت ترجمة هذا الكتاب ليشكل مرجعاً شاملاً لمن يريد الغوص أكثر في جنبات هذا العالم الحرّ، عالم جنو/لينكس، آملين أن يفيد الدارسين في عالمنا العربي وأن يشكل لبِنة بناء في صرح مجد أمتنا التليد.

الاشتراك

نشجّع مستخدمي المواد التعليمية للأكاديمية على تقديم تغذية راجعة واقتراحات لتحسين الكتاب. هناك مساحة مخصصة متوفرة على موقع الأكاديمية للتغذية الراجعة. سيتم أخذ هذه المدخلات بعين الاعتبار للإصدارات اللاحقة. إضافة إلى ذلك، ترحب الأكاديمية بكل من يرغب باستخدام أو توزيع هذه المواد، إضافة إلى عمل ترجمات وإصدارات جديدة.

لمعاومات معيّنة ومحدثة عن الكتاب – بما في ذلك الترجمات والهيئات الأخرى – راجع:

http://ftacademy.org/materials/fsm/2. لمزيد من المعلومات، وللتسجيل في برنامج الأكاديمية، نرجو مراجعة موقع الأكاديمية: http://ftacademy.org.

أرجو من كل قلبي أن يساعدك هذا الكتاب في عملية التعليم الذاتي، وأن يساعدك على مساعدة الآخرين في تعليمهم كذلك. وآمل أن أراك في حراك المعرفة الحرّة والتقنية الحرّة.

أرجو لك الاستمتاع بما تتعلمه!

بتصرّف، ووتر تبنز

Wouter Tebbens

رئيس مؤسسة المعرفة الحرّة

مدير أكاديمية التقنية الحرّة

المحتويات

مقدمة إلى

نظام التشغيل

جنو/لينكس

د. جُسِـــپ جُبرا إستيڤَ

مقدمة

لم يعد نظام جنو/لينكس شيئاً جديداً، فلهذا النظام نطاق واسع من المستخدمين، وهو مستخدم في معظم بيئات العمل.

يعود تاريخ هذه الأنظمة إلى شهر آب/أغسطس (8) عام 1991، عندما أعلن طالب فنلندي اسمه لينوس تورفالدز على قائمة إخباريّة أنه أنشأ نظام تشغيله الخاص، وأنه يقدمه إلى مجتمع المطورين لاختباره واقتراح تحسينات لجعله أكثر قابلية للاستخدام. كانت هذه بداية أساس (أو نواة kernel) النظام، والتي صار اسمها لينكس لاحقاً.

كانت مؤسسة البرمجيات الحرّة FSF منذ 1984 تنتج ضمن مشروعها المسمى جنو GNU برامج يمكن استخدامها بحريّة. وقد أوضح ريتشارد ستولمان (عضو FSF) أن البرمجيات الحرّة هي تلك التي يمكن لمصدرها البرمجي (source code) أن يؤخذ ويُدرس ويعدَّل وينشر دون أن تجبر على الدفع مقابل ذلك. لا يرتكز العمل ضمن هذا النموذج على إخفاء المصدر، بل على البرامج الإضافيّة المكمّلة، وتعديل البرنامج ليناسب احتياجات زبون معيّن، والخدمات الإضافيّة كالصيانة وتدريب المستخدمين والدعم الفنّي، سواء كان ذلك عبر مصادر تعليمية كالكتب وأدلة الاستخدام أو عبر الدورات التدريبيّة.

التركيبة الناتجة من مزج برمجيات جنو ونواة لينكس هي أنظمة جنو/لينكس الموجودة حاليًّا. حركة البرمجيات الحرة، القائمة الآن على مؤسسات مثل FSF والشركات التي تنتج توزيعات جنو/لينكس المختلفة (ردهات، ومادريفا، وسوزي، وغيرها ...)، إضافة إلى أن شركات كبرى تقدم دعماً، مثل HP و IBM و Sun قد أعطت دفعة لأنظمة جنو/لينكس لتجعلها في مستوىً يسمح لها بمنافسة والتفوق على حلول مغلقة ومملوكة.

لم تعد أنظمة جنو/لينكس جديدة. لقد بدأت برمجيات جنو في منتصف الثمانينيات، ونواة لينكس في بداية التسعينات. وقد بني لينكس على تقنيات أنظمة يونكس المجربة وذات التاريخ الطويل ذي الـ 40 عاماً.

سنراجع في هذه الوحدة بعض الأفكار العامة لحركتي البرمجيات المفتوحة والحرّة، إضافة إلى تاريخ نظام لينكس وأصله المشترك مع يونكس، والذي استفاد من أبحاثه التي تخطت ثلاثين عاماً في أنظمة التشغيل.

1 البرمجيات الحرة والمصدر المفتوح

لدينا عدد من النماذج المختلفة للبرمجيات والتي تشترك في أفكار معينة، ومنها حركة البرمجيات الحرة وحركة المصدر المفتوح.

> إن كون البرنامج مفتوح المصدر يعني ضمنيًّا أن فكرته الأساسيّة هي إمكانيّة الوصول إلى مصدره البرمجيّ، وتعديله وإعادة توزيعه ضمن شروط معينة وحسب ما هو مذكور في رخصة معينة للمصدر المفتوح تحدد السياق القانونيّ لهذه العمليّة.

على النقيض من البرمجيات المغلقة، والتي تغلق فيها الشركة المنتجة المصدر البرمجي وتخفيه وتمنع غيرها من استخدامه، دون إتاحة أيّة إمكانيّة لأيّ تعديل أو تغيير ما لم تكن هذه التعديلات أو التغييرات تمّت من طرف المنتج نفسه؛ يتيح المصدر المفتوح ما يلي:

1) الوصول إلى المصدر البرمجيّ، سواء كان هذا لدراسته (كمصدر مثالي للتعليم)، أو لتعديله أو تصحيح الأخطاء أو تطويعه أو إضافة المزيد من المزايا.

2) برمجيات مجانية: في العادة يمكن الحصول على البرمجيات مفتوحة المصدر سواء كملفات ثنائية أو كود (code) برمجي مجاناً، أو بمبلغ زهيد يكفي لتغطية تكاليف التحزيم والتوزيع والقيمة المضافة.

3) معايير قياسية تمنع الاحتكار الذي تقوم به شركات البرمجيات المغلقة، مما يجنب المستخدمين الاعتماد على شركة برمجيات معينة كخيار وحيد؛ هذا أكثر أهمية بالنسبة للمؤسسات الكبيرة، سواء كانت شركة أو حكومة، حيث لا يمكنها أن تضع نفسها بين يدي حلّ محدّد ووحيد وتعتمد عليه بشكل حصري، أو بالأحرى عليها أن لا تفعل ذلك.

4) نموذج تقدم لا يعتمد على إخفاء المعلومات، وإنما على مشاركة المعرفة (كالمجتمع العلميّ) مما يتيح إمكانيّة التقدّم بسرعة أكبر، وبجودة أعلى وذلك لأنه يعتمد على مشاركة المجتمع بالرأي، وليس على مزاج الشركات التي تطور

برمجيات مملوكة.

إنشاء برامج وتوزيعها مع مصدرها البرمجي ليس شيئاً جديداً. لقد كانت الأمور تتم بهذه الطريقة منذ بداية تكنولوجيا المعلومات والإنترنت. لكن مفهوم "مفتوح المصدر" نفسه وكتابة المسودة الأولية للشروط الواجب توفرها ليعتبر البرنامج مفتوح المصدر يعود لمنتصف عام 1997.

أعلن إيريك ريموند وبروس بيرينز عن الفكرة. لقد كان رايموند مؤلف مقالة "الكاتدرائيّة والبازار"[1] والتي ناقشت تقنيات التطوير التي اتبعها مجتمع لينكس بقيادة لينوس تورفالدز ومجتمع جنو التابع لمؤسسة البرمجيات الحرة FSF بقيادة ريتشارد ستولمان. وكان بروس بيرنز قائد مشروع دبيان الذي كان يعمل على إنشاء توزيعة جنو/لينكس مبنية حصرياً على برمجيات حرّة.

ملاحظة:
من أهم المجتمعات ذات العلاقة بهذا الموضوع مؤسسة البرمجيات الحرة ومشروع البرمجيات جنو التابع لها، ومجتمع المصدر المفتوح ومشروعه الأساسيّ "لينكس". جنو/لينكس هو ناتج دمج عمل هذين المجتمعين.

يكمن الفرق الأساسيّ بين هذين المجتمعين في تعريف البرمجيات الحرّة ومفتوحة المصدر.

مؤسسة البرمجيات الحرة مؤسسة غير ربحيّة أنشأها ريتشارد ستولمان الذي يؤمن بأن علينا أن نضمن بأن تكون البرمجيات في متناول أيدي الجميع، دون أيّة تكاليف، ويمكن الوصول إليها بحرّيّة لتستخدم بالطريقة التي يراها المستخدم مناسبة. أثارت كلمة free تحفظات بعض الشركات، حيث يمكن للكلمة أن تعني "مجاني" (دون تكاليف أو مقابل)، أو أن تعني "حرّ" (ليس تحت سيطرة أحد). التمست مؤسسة البرمجيات الحرة معاني الاثنتين، لكن تسويق هتين الفكرتين للشركات كان صعباً، حيث كان السؤال الأساسيّ: "كيف يمكننا الحصول على مال بهذا؟!" . أتت الإجابة من مجتمع لينكس بقيادة لينوس تورفالدز عندما تمكّن هذا المجتمع من الحصول على شيء لم تكن مؤسسة البرمجيات الحرة ومشروعها جنو يملكانه بعد: نظام تشغيل حرّ ومجّانيّ بمصدر برمجيّ متاح. عند تلك النقطة، قرر المجتمع توحيد النشاطات المختلفة ضمن حركة البرمجيات الحرة تحت اسم جديد:

---

1 كان قد تم طرح "الكاتدرائيّة والبازار" للترجمة منذ زمن وترجمت أجزاء منه لا أدري أكتملت ترجمته أم لا. يمكن البحث عنه في الإنترنت وفي منتدى مجتمع لينكس العربي. يمكن الوصول إلى النصّ الأصليّ من هذا الرابط: http://www.catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s03.html

البرمجيات مفتوحة المصدر.

تم تسجيل المصدر المفتوح كعلامة تجارية لإعطاء شهادات للبرمجيات التي تلتزم بمعاييرها. هذه الخطوة لم تنل رضى الجميع، وهناك انقسام أو جدل بين المجموعتين (المصدر المفتوح ومؤسسة البرمجيات الحرة)، رغم التشابه الكبير بينهما.

إلى حد ما، وحسب تعبير البرمجيات الحرة، المصدر المفتوح خطوة خاطئة، وذلك لأنها تعني بيع مبادئها للسوق، تاركة الباب مفتوحاً لتحويل البرمجيات الحرة إلى مملوكة. أما مؤيدو المصدر المفتوح فيرون فيها فرصة للترويج لبرمجيات ما كان ليستخدمها إلا قلة من المستخدمين، حيث يُمكّننا نشرُ هذه البرمجيات بهذه الطريقة - حتى بين الشركات التي ترغب بالمساهمة في المصدر المفتوح - من إيجاد قوة كافية لتحدي البرمجيات المملوكة.

مع هذا، فالفكرة الأساسيّة من وراء كلتي الحركتين هي زيادة استخدام البرمجيات الحرة، مما يوفر بديلاً للبرمجيات المملوكة التي تأمل الشركات الكبرى في نشرها. تبقى هذه الاختلافات بعيدة عن الجانب العمليّ التطبيقيّ.

بعد الخروج بفكرة مجتمع المصدر المفتوح، صار تحديد المعايير التي على البرمجيات اتباعها لتصنف كبرمجيات حرّة حاجة أساسية. كان علينا أن نبنيه على التعريف الذي كتبه بروس بيرينز للمصدر المفتوح في حزيران/يونيو 1997 كرد على مطوري توزيعة دبيان جنو/لينكس والتي قامت مؤسسة مبادرة المصدر المفتوح Open Source Initiative – OSI بإجراء تعديلات طفيفة عليها لاحقاً. إنّ مهمة مبادرة المصدر المفتوح هي التحكم بتعريف المصدر المفتوح وتراخيصه.

ملاحظة

يعتمد المصدر المفتوح على تعريف معروف للعموم ويعتمد عليه كأساس لكتابة تراخيص البرمجيات مفتوحة المصدر.

كتلخيص وشرح موجز للتعريف: على البرمجيات مفتوحة المصدر، أو البرمجيات متاحة الأكواد تحت أحد تراخيص المصدر المفتوح، أن توفّر المتطلبات التالية:

1) يمكن للبرمجية أن تُنسَخ أو توهَب أو تُعطى لأيّ طرف ثالث دون الحاجة للدفع مقابل ذلك.

2) على البرنامج أن يتيح المصدر البرمجي وأن يسمح بتوزيع هذا المصدر البرمجي والصيغة التطبيقية المجمّعة والجاهزة

للاستخدام. أو وجوب وجود إمكانية معروفة جيداً ومنشورة للحصول على المصدر البرمجي في جميع الأحوال، كالتنزيل من الإنترنت على سبيل المثال. لا يسمح بأي نوع من الأكواد المُبهمة أو الوسيطة الموضوعة بشكل متعمد، وعلى الرخصة أن تضمن إمكانية عمل تعديلات.

3) على الرخصة أن تسمح بالتعديلات والأعمال المشتقّة، وعليها ان تسمح بتوزيع هذه التعديلات والأعمال المشتقّة ضمن نفس رخصة العمل الأصليّ، مما يسمح للكود الأصلي أن يعاد استخدامه.

4) يمكن اشتراط صحة المصدر البرمجي للمؤلف، أي أنه يمكن اشتراط كون التعديلات على شكل ترقيعات patch يمكن تنفيذها على المصدر الأصليّ، أو أن يكون لزاماً عليها أن تحمل أسماء أو أرقام إصدارات مختلفة عن المشروع الأصليّ. هذا يحدّد أيّة تعديلات يمكن نسبتها للمؤلف. يعتمد هذا البند على الترخيص المستخدَم.

5) يجب ان لا يحوي الترخيص تمييزًا ضد شخص أو مجموعة. يجب عدم تقييد الوصول إلى البرمجية. يمكن في بعض الحالات أن تكون هناك قيود قانونيّة، كما في قانون الولايات المتحدة الأمريكية لتصدير التقنية إلى دول أخرى. إذا كان هناك قيود من هذا النوع، فيجب ذكرها.

6) عدم وجود تمييز ضدّ أهداف أو مجالات عمل. يمكن للبرنامج أن يستخدم في أيّ مجال ولأيّة أهداف كانت، حتى وإن لم تكن أعدّت لهذا الغرض. يسمح بالاستخدام التجاريّ، وليس لأحد أن يمنع استخدامه لأيّة أغراض تجارية.

7) تنطبق الرخصة على كلّ من يصلهم البرنامج.

8) إذا كان البرنامج جزءًا من منتج أكبر، فعليه أن يستخدم نفس الرخصة. تضمن هذه النقطة عدم أخذ أجزاء من برمجيات مفتوحة المصدر لعمل برمجيات مملوكة. في حالة استخدام أجزاء من برمجيات مفتوحة المصدر في برمجيات مملوكة، فيجب ذكر ذلك وتحديد هذه الأجزاء.

9) على الترخيص أن لا يقيّد أي تكامل أو تجميعة للبرمجيات، وبكلمات أخرى، كون البرنامج مفتوح المصدر جزءًا من

تجميعة معيّنة يجب أن لا يمنع وجود أيّ منتج آخر ناتج عن تجميعة مختلفة لهذا البرنامج مع برامج أخرى. هذه النقطة مثيرة للجدل، حيث تبدو مناقضة لسابقتها، حيث تشترط هذه النقطة إمكانيّة أخذ أيّة برمجيات مفتوحة المصدر وإضافتها إلى برمجيات أخرى دون التأثير على ترخيصها (كضمها إلى برمجيات مملوكة)، لكنها تشترط أيضاً ذكر وجود أجزاء للبرمجية مفتوحة المصدر.

10) على الرخصة أن تكون محايدة تقنيًّا. فمثلاً، عليها أن لا تكون مرتبطة بأجهزة أو أنظمة تشغيل معيّنة. يمكن اشتراط وسائل معيّنة لنشر هذه البرمجيات أو وضع استثناءات يمنع النشر عبرها. فمثلاً، يمكن حصر طرق نشر برمجية معينة بالأقراص المضغوطة و FTP وخوادم الشَنكبوتيّة.

تعريف المصدر المفتوح هذا ليس رخصة بحد ذاته لكنه مجموعة من المعايير المطلوبة في أيّ ترخيص ليعتبر ترخيصًا للمصدر المفتوح.

يجب على الترخيص أن يلتزم بالمعايير المذكورة أعلاه ليعتبر البرنامج المرخّص به مفتوح المصدر. مسؤوليّة مبادرة المصدر المفتوح OSI هي التأكد من مطابقة الترخيص لهذه الشروط والمعايير. يمكنك إيجاد قائمة بالتراخيص مفتوحة المصدر على صفحة موقع مبادرة المصدر المفتوح، والتي ستجد بها رخصة جنو العامّة GNU General Public License – GPL التي تعدّ أشهر وأكثر هذه التراخيص استخداماً.

يمكن للبرمجيات المرخصة بـ GPL أن تُنسَخ وتُعَدَّل، لكن يجب ان تتاح هذه التعديلات للعموم ضمن نفس الترخيص، مما يمنع اختلاط أكواد مفتوحة بأخرى مملوكة مغلقة، وبهذا يتجنّب سيطرة الأكواد المملوكة على الأكواد مفتوحة المصدر. لدينا أيضاً ترخيص LGPL المشابه له، إلا أنه يسمح بتضمين البرمجيات المرخّصة به ضمن أخرى مملوكة. من الأمثلة المعروفة مكتبة Linux C المرخصة بترخيص LGPL، والتي لو كانت مرخصة بـ GPL لما أمكن تطوير سوى برمجيات مفتوحة المصدر عليها، لكن كونها مرخصة بـ LGPL يجعل بالإمكان استخدامها لتطوير برمجيات مملوكة أيضاً.

العديد من مشاريع البرمجيات الحرة – أو حتى المشاريع التي تأتي بجزء مفتوح المصدر وآخر مملوك – لديها تراخيصها الخاصّة بها: أباتشي (مبنية على BSD)، وموزيلا (MPL، و NPL من Netscape) ... إلخ. عندما نريد عمل برنامج مفتوح

المصدر فنحن مخيرون بين كتابة ترخيص خاصّ بنا يتوافق مع المعايير المذكورة آنفاً، أو أن نختار إحدى الرخص الموجودة مسبقاً والتي تتوافق مع هذه المعايير، في حالة استخدام أجزاء مرخصة بـ GPL، فنحن ملزمون باستخدام هذه الرخصة للبرنامج بأكمله.

الآن، وبعد أن درسنا مفاهيم البرمجيات مفتوحة المصدر وتراخصيها، علينا أن نرى الآن إلى أيّ مدى يمكن لشركة ما أن تستفيد من إنتاج برمجيات مفتوحة المصدر. إذا لم تكن هذه المفاهيم جذّابة للشركات، فسنخسر زبوناً محتملاً وواحدة من الشركات الرائدة في إنتاج البرمجيات مفتوحة المصدر في وقت واحد!

المصدر المفتوح جذاب للشركات، مع نموذج ربحيّ يركّز على القيمة المضافة للمنتج. يتيح المصدر المفتوح فوائد جذّابة عديدة تهمّ الشركات، منها:

1) بالنسبة للشركات المطورة للبرمجيات، يوحي هذا بوجود مشكلة: كيف يمكن الحصول على المال دون بيع منتَج؟ يُنفّق كثير من المال على تطوير برنامج، ومن ثمّ على الشركة استعادة هذا المبلغ مع عائد ربح عليه. لا توجد إجابة واضحة على هذا التساؤل، ولا حتّى أيّ نوع من أنواع البرمجيات، حيث يعتمد العائد على نوع البرمجيات التي يمكنها جلب عائد يتخطى مجرد بيع البرنامج. عادة ما يتم عمل دراسة لتحدد إذا ما كان فتح مصدر البرنامج سيعود بالنفع على المنتج أم لا (في أغلب الحالات سيكون إيجابيًّا)، على فرض أن تكلفة التطوير ستكون أقل (حيث سيساهم فيه المجتمع)، وكذلك ستقل تكلفة الصيانة والمتابعة وإصلاح العلل (سيساعد المجتمع في هذا بشكل سريع)، آخذين بعين الاعتبار عدد المستخدمين الذي سيجذبهم كون البرنامج مفتوح المصدر، واحتياجات هؤلاء المستخدمين لخدمات الدعم الفنّي والتوثيق. إذا كانت نتيجة الموازنة إيجابيّة، فسيكون من المنطقي فتح مصدر البرنامج والاستغناء عن عائدات البيع.

2) زيادة عدد المستخدمين.

3) الحصول على مرونة أكبر في التطوير، فكلما زاد عدد المعنيّين، زاد عدد الأشخاص الذين يمكنهم اكتشاف الأخطاء.

4) سيأتي العائد غالبًا من الدعم الفنّيّ وتدريب المستخدمين والصيانة والمتابعة.

5) على الشركات المستخدمة للبرمجيات أن تأخذ بعين الاعتبار عددًا من المعامِلات قبل اختيار برنامج لإدارة مهامها، كالأداء والموثوقيّة والأمن وإمكانيّة التوسّع والتكلفة الماديّة. وعلى الرغم من أن المصدر المفتوح قد يبدو خياراً بديهيًّا من ناحية التكلفة، فعلينا أن نقول أيضًا أن هناك برمجيات مفتوحة المصدر يمكنها مجابهة (وحتى تخطّي) برمجيات مملوكة في أيّ من المعامِلات الأخرى. علينا أيضًا أن ننتبه عند اختيار أنظمة مملوكة من مُصَنِّع واحد؛ لا يمكننا الاعتماد على هذه الأنظمة وحدها (يمكننا تذكُّر حالات مثل صيغة الفيديو بيتا من Sony مقابل VHS، أو معماريّة MicroChannel للأجهزة الشخصيّة PC من IBM). علينا تجنُّب المحتكرين والمخاطر المتعلّقة بهم: كانعدام المنافسة في السعر، والخدمات عالية التكلفة، والتكلفة العالية للصيانة، وقلّة (أو انعدام) الخيارات ...إلخ.

6) أما على مستوى الاستخدام الشخصيّ، فتوفر طيفاً واسعاً من البرمجيات المخصصة للمستخدمين العاديين، حيث أُعِدَّت ونُفذت كثير من هذه البرمجيات على أيدي أناس احتاجوا للقيام بنفس المهام، لكنهم لم يتمكّنوا من إيجاد البرنامج المناسب. في حالة المستخدمين المنزليين يعدّ عامل التكلفة عادة عاملاً مهمًّا جدًّا، لكن التناقض الذي يحصل هو أنّ هؤلاء المستخدمين يميلون أكثر إلى استخدام برمجيات مملوكة. عادة ما يستخدم المستخدمون المنزليون نسخاً غير قانونيّة لبرمجيات مملوكة؛ تشير إحصائيّات حديثة إلى أنّ 60-70% من البرمجيات المملوكة للاستخدام المنزلي نسخ غير قانونيّة. يشعر المستخدمون أنه بمجرد امتلاكهم جهاز حاسوب شخصيّ، فإنّهم مخوّلون في بعض الدول لاستخدام البرمجيات (المملوكة) له. نحن نتعامل في هذا مع نسخ غير قانونيّة، ورغم أن هؤلاء قد لا يكونون عوقِبوا على ذلك، فيمكن أن يعاقبوا عليه في يوم من الأيام، أو أنه يتمّ السيطرة عليهم بأنظمة تراخيص وتفعيل للمنتجات. لهذا أيضاً أثر سلبيّ غير مباشر على البرمجيات الحرّة، لأنه إذا كان المستخدمون يستخدمون هذه البرمجيات بكثرة، فسيجبر هذا المستخدمين الذين يرغبون بالتواصل معهم - سواء كان هؤلاء المستخدمين بنوكًا أو شركات أو مؤسسات حكوميّة أو ما إلى ذلك - على استخدام هذه المنتجات المملوكة، وسيكون لزاماً على هؤلاء المستخدمين الدفع مقابل تراخيص هذه البرمجيات. إنّ من أهم ما تقوم به البرمجيات الحرّة هو المنازعة للحصول على المستخدمين المنزليين.

7) وفي النهاية، الحكومات تحديداً تحصل على فوائد هامّة من البرمجيات مفتوحة المصدر، حيث توفّر برمجيات ذات جودة عالية بأسعار سخيفة إذا ما قورِنَت بتكاليف تراخيص البرمجيات المملوكة. إضافة إلى هذا، يمكن للبرمجيات مفتوحة المصدر أن تتضمّن قيماً ثقافيّة بسهولة (لكلّ دولة أو مجتمع على حدة)، كاللغة على سبيل المثال. هذه الحالة الأخيرة تسبب معضلة، حيث يرفض بعض منتجي البرمجيات المملوكة تطويع برمجياتهم لبعض المناطق، كالدول الصغيرة التي لديها لغة خاصة بها، وبعضها تطلب الدفع مقابل ذلك.

فلنراجع بعضاً من تاريخ نظام يونكس، كونه سَلَف نظام جنو/لينكس. في البداية اعتُبِرَ نظام لينكس تقليداً لنظام مينيكس (نظام يونكس تعليمي للأجهزة الشخصية)، واستخدم بعض الأفكار التي طوّرتها أنظمة يونكس المملوكة؛ لكنه طوّر كنظام مفتوح المصدر للعمل على أجهزة PC المنزليّة. في هذا القسم المختصّ بنظام يونكس، والنظام التالي المختص بنظام جنو/لينكس، سنرى كيف قادنا هذا التطور إلى أنظمة جنو/لينكس الحاليّة القادرة على مواجهة أيّ نظام يونكس مملوك والمتوفرة لعدد هائل من المعماريّات، من الأجهزة الشخصيّة وحتى الحواسيب الخارقة.

يمكن استخدام لينكس على طيف واسع من الأجهزة. يمكننا إيجاد العديد من الحواسيب الخارقة التي تعمل بنظام جنو/لينكس في قائمة TOP500 (على http://top500.org)؛ فجهاز MareNostrum في مركز برشلونة للحوسبة الخارقة شبكةٌ عنقوديّة بـ 10240 معالج CPU من نوع PowerPC صممتها IBM وتعمل بنظام تشغيل جنو/لينكس تمّ تطويعه لتغطية متطلّبات هذه الأجهزة. يمكننا أن نلاحظ أن النسبة الكلّيّة للحواسيب الخارقة التي تعمل بأنظمة جنو/لينكس في هذه القائمة تصل إلى 75%.[2]

بدأ يونكس في 1969 – مما يجعله ذو تاريخ يتعدى الأربعين عاماً – وكان ذلك في معامل Bell Telephone Labs – BTL، التابعة لشركة AT&T الأمريكيّة. انحدرت أنظمة يونكس من مشروع سمّي MULTICS، كان قد تمّ تصميمه لإنشاء نظام تشغيل يمكن أن يجعل حاسوباً كبيراً يدعم آلاف المستخدمين في نفس الوقت، وكان يعمل على المشروع حينها مختبرات BLT و شركة General Electrics وجامعة ماساشوستس MIT. لكن المشروع فشل لأن طموحه كان أكبر من الإمكانيّات المتاحة في ذلك الوقت.

عندما تمّ إهمال المشروع، وجد مهندسان من BTL كانا يعملان على مشروع MULTICS، وهما كين تومسون ودينيس ريتشي حاسوباً من نوع DEC PDP7 لم يكن أحد يستخدمه، ولم يكن عليه سوى مجمّع وبرنامج إقلاع بسيط (

---

2 تعود هذه الإحصائية لما قبل تأليف الكتاب الأصلي عام 2009، ففي آخر إحصائيّة سبقت ترجمة هذا الكتاب (إحصائيّة حزيران/يونيو 2012)، وصل عدد الحواسيب الخارقة التي تعمل بنظام لينكس في هذه القائمة إلى 462 جهازاً، بما نسبته 92.4% من هذه الحواسيب، والبقيّة تعمل بأنظمة يونكس أخرى أو بأنظمة هجينة.

Loading Program). طور تومسون وريتشي أجزاء من يونكس كاختبار (غالباً في أوقات فراغهما)، ومجّمِعاً (للغة الآلة)، والنواة الأولية لنظام التشغيل.

في نفس العام، فكر تومسون بعمل نظام ملفات لهذه النواة، بطريقة تسمح للملفات أن تخزَّن بطريقة منظَّمة في نظام أدلّة أو مجلّدات متفرّعة. بعد العديد من المناظرات النظريّة التي امتدت لشهرين، تم تنفيذ النظام خلال بضعة أيّام فقط. مع التقدم في تصميم النظام وانضمام المزيد من المهندسين إلى المشروع، صار الجهاز الأصليّ صغيراً جدًّا على النظام، وفكّروا بطلب واحد جديد (كان يكلف حينها حوالي مئة ألف دولار أمريكيّ، مما يجعله استثماراً ضخماً). كان عليهم اختلاق عذر (حيث كان يتم تطوير نظام يونكس في وقت الفراغ)، فقالوا أنهم يريدون إنشاء محرر نصوص جديد (وهو من البرامج التي كانت تدرّ أرباحاً في ذلك الوقت)، ولهذا أعطوا موافقة لشراء جهاز PDP11.

> يعود تاريخ يونكس إلى 1969، وله أكثر من 30 عاماً من التطوير التقنيّ والاستخدام على كلّ أنواع الأنظمة.

عندما وصل الجهاز، أُعطوا فقط المعالج والذاكرة، ولم يعطوا القرص أو نظام التشغيل. لم يقدر تومسون على الانتظار، فصمم قرصاً في الذاكرة RAM، واستخدم نصف الذاكرة كقرص والنصف الآخر لنظام التشغيل الذي كان يعمل على تصميمه. عندما وصل القرص، أكملوا العمل على نظام يونكس وعلى برنامج معالجة النصوص الموعود (الذي استخدموه كعذر). نجح برنامج معالجة النصوص (وكان Troff، لغة محرّر استخدمت لاحقاً لإنشاء دليل استخدام يونكس المسمى man pages). بدأت مختبرات BTL استخدام النسخة الأولية من يونكس مع محرر النصوص الجديد، مما جعل BTL المستخدم الأول لنظام يونكس.

في ذلك الوقت، بدأت طلائع فلسفة يونكس بالبزوغ:

- كتابة برامج تفعل شيئًا واحدًا، وتقوم به كما يجب.
- كتابة برامج لتعمل سويًّا.

- كتابة برامج للتعامل مع السلاسل النصّيّة.

ومن المميزات الهامة في يونكس أنه كان من أول الأنظمة المستقلّة عن معماريّة العتاد، مما جعله قابلاً للتنقل بين عدد كبير من المعماريّات المختلفة.

في تشرين الثاني/نوفمبر 1971، وبسبب وجود مستخدمين من خارج المعمل، دعت الحاجة لوجود دليل مبرمجي يونكس، والذي صمّمه تومسون وريتشي. في الإصدار الثاني (حزيران/يونيو 1972) والمعروف بـ V2 (والذي تم فيه تعديل رقم إصدار دليل الاستخدام ليتوافق مع رقم إصدار يونكس) قيل بأن عدد مرات تثبيت يونكس وصلت إلى عشرة، واستمر الرقم بالارتفاع إلى أن وصل خمسين تثبيتاً في الإصدار V5.

في نهاية عام 1973، تقرّر عرض النتائج في مؤتمر عن أنظمة التشغيل. وكنتيجة لذلك، طلب عدد من مراكز تكنولوجيا المعلومات والجامعات نسخاً من يونكس.لم توفر AT&T صيانة أو دعماً ومتابعة ليونكس، مما أدّى إلى وجوب توحّد مستخدمي يونكس ومشاركة المعرفة بينهم بما يعرف بمجتمع مستخدمي يونكس. بدأ المستخدمون بمشاركة أفكارهم ومعلوماتهم عن البرامج والعلل وما إلى ذلك. لقد أنشأوا منظّمة أطلقوا عليها اسم USENIX، والتي تعني مستخدمي يونكس users of UNIX. وقد حضر اجتماعهم الأول في 1974 اثنا عشر شخصًا.

لقد كانت جامعة كاليفورنيا في بيركلي – وهي المكان الذي درس فيه تومسون – من الجامعات التي حصلت على ترخيص يونكس. وفي عام 1975، عاد تومسون إلى بيركلي كمدرّس وأحضر معه أحدث إصدار يونكس. انضم إليه طالبان حديثا التخرّج وهما تشاك هيلي و بيل جوي (والذي كان نائب رئيس شركة Sun Microsystems عندما أُلّف هذا الكتاب)، وعملوا معاً على تنفيذ يونكس.

كانت المحررات من الأمور التي استاؤوا منها؛ قام جوي بإنشاء محرر نصوص جيّد أطلق عليه اسم EX إلى أن تم تحويله إلى VI،وهو محرر نصوص مرئي ملء الشاشة. وقد طوّر الإثنان مصنِّفًا للغة جافا أضافاه إلى يونكس. كان هناك طلب جيد على هذا الإصدار من يونكس، وبدأ جوي يصدره كتوزيعة برمجيات بيركلي BSD.

كان لـ BSD حينها ترخيصٌ معيّنٌ مقابل ثمنه: قيل بأنه يتناسب مع تكلفة النسخ والتوزيع في ذلك الوقت. لهذا أصبح

المستخدمون يعملون تعديلاتهم، ويضيفون مزايا، ويبيعون نسخهم المعدّلة، وبعد مدّة معينة كانت تضاف هذه التعديلات إلى الإصدار الجديد من BSD.

لجوي مساهمات أخرى أيضاً في برنامجه vi، فقد إضاف إليه ميزة التحكّم بالطرفيات النصّيّة مما جعل المحرر مستقلاً عن الطرفيّة التي يعمل عليها؛ لقد أنشأ نظام TERMCAP كواجهة طرفيات شاملة مع متحكّمات لكلّ نوع من الطرفيّات، والتي مكنت البرنامج من العمل بغض النظر عن الطرفيات باستخدام هذه الواجهة.

كانت الخطوة التالية تطويعه للعمل على معمار يات مختلفة. حتى عام 1977، كان من الممكن تشغيله على أجهزة PDP فقط؛ وفي ذلك العام تمّ تطويع البرنامج للعمل على الأجهزة المتوفّرة في ذلك الوقت كـ Interdata و IBM. كان إصدار يونكس V7 عام 1979 أول إصدار محمول. وفر هذا الإصدار عدداً من المزايا المتقدّمة، فقد حوى: ,awk, lint, make uucp؛ لقد حوى دليل الاستخدام في ذلك الوقت 400 صفحة، مع ملحقين كلّ منهما 400 صفحة. لقد حوى أيضاً مُصرِّف لغة C الذي صمّمه كيرنيغان وريتشي في معامل BTL، والذي أُنشئ لإعادة كتابة معظم أجزاء يونكس. لقد تمّ بناؤه في البداية باستخدام المجمّع، ثم نقله إلى لغة C مع بقاء الأجزاء المعتمدة فقط على المعماريّة مكتوبة بالمجمّع. لقد تمّ أيضًا إضافة صدفة محسّنة (bourne shell) وأوامر مثل: find, cpio, expr.

لقد بدأت صناعة يونكس بالنمو، وبدأت إصدارات وأنواع مختلفة من يونكس تظهر: كنظام Xenix المشترك بين مايكروسوفت – والتي عملت في أيامها الأولى على إصدارات يونكس – وشركة SCO، والذي كان موجّهاً لأجهزة Intel 8086 (أول أجهزة IBM المكتبية)، وإصدارات جديدة من BSD كذلك ...

رغم هذا، فقد ظهرت مشكلة جديدة عندما أدركت AT&T أن يونكس نظام تجاريّ قيّم، فأضحى الإصدار V7 متاحاً برخصة تمنع تدريسه في المؤسسات التعليميّة وذلك لحماية أسراره التجاريّة. كانت العديد من الجامعات حتى ذلك الوقت تستخدم المصدر البرمجي لنظام يونكس لتدريس أنظمة التشغيل، لكنهم توقفوا عن ذلك وصاروا يدرّسون الجانب النظري فقط.

لكن تمكّن البعض من إيجاد طريقة لحلّ هذه المشكلة. ففي أمستردام – عاصمة هولندا – قرّر أندرو تانيبوم – وهو كاتب مرموق لكتب نظرية عن أنظمة التشغيل – أن يكتب نظام تشغيل جديداً متوافقاً مع يونكس دون استخدام أيّ سطر

برمجي من أكواد AT&T، وسمّى ذلك النظام Minix، وقد استَخدَم هذا النظام في ما بعد، وتحديداً عام 1991، طالبٌ فنلنديّ لإنشاء إصدار خاص به من يونكس أسماه لينكس.

وفي ذلك الوقت، قرر بيل جوي – الذي ما برح يطوّر نظام BSD في بيركلي (والذي كان قد وصل إلى الإصدار 4.1) – أن ينتقل إلى شركة جديدة اسمها SUN Microsystems، والتي أنهى فيها العمل على BSD 4.2، الذي عُدِّلَ فيما بعد لإنشاء نظام يونكس الخاص بشركة SUN، والذي سُمِّي SunOS، وذلك قرابة عام 1983. بدأت كلّ شركة منذ ذلك الحين بتطوير نظام يونكس الخاص بها: فقد طورت IBM نظام AIX، وشركة DEC طورت Ultrix، و HP طورت HPUX، وكل من مايكروسوفت و SCO طورتا Xenix ...إلخ. منذ عام 1980، والذي بدأ فيه يونكس كمغامرة تجارية، أطلقت AT&T إصداراً أخيراً أسمته "نظام يونكس الخامس" UNIX System V، والذي يشار إليه اختصاراً SV[3]. وقد بُنيت كلُّ أنظمة يونكس الحديثة إما عليه أو على BSD 4.x. لقد تمت مراجعة SV عدة مرات، فمثلاً، اعتُبِر الإصدار الرابع من أهم إصداراته. كانت نتيجة هذه الإصدارات الأخيرة أنه قد تم تعديل أنظمة يونكس المختلفة لتصير متوافقة مع بعضها تقريباً؛ فقدكانت في الغالب نسخاً من System V الإصدار الرابع من AT&T أو BSD من بيركلي معدّلة من طرف مصنّعي العتاد. كان يحدد بعض المصنعين ما إذا كان نظام يونكس الذي يوفرونه من نوع SV أو BSD، لكنه كان في الحقيقة مزيجاً يحوي أجزاء من النوعين. تمّ فيما بعد وضع معايير لأنظمة يونكس لجعلها متناغمة مع بعضها، من ضمنها معايير IEEE POSIX و UNIX97 و FHS وغيرها.

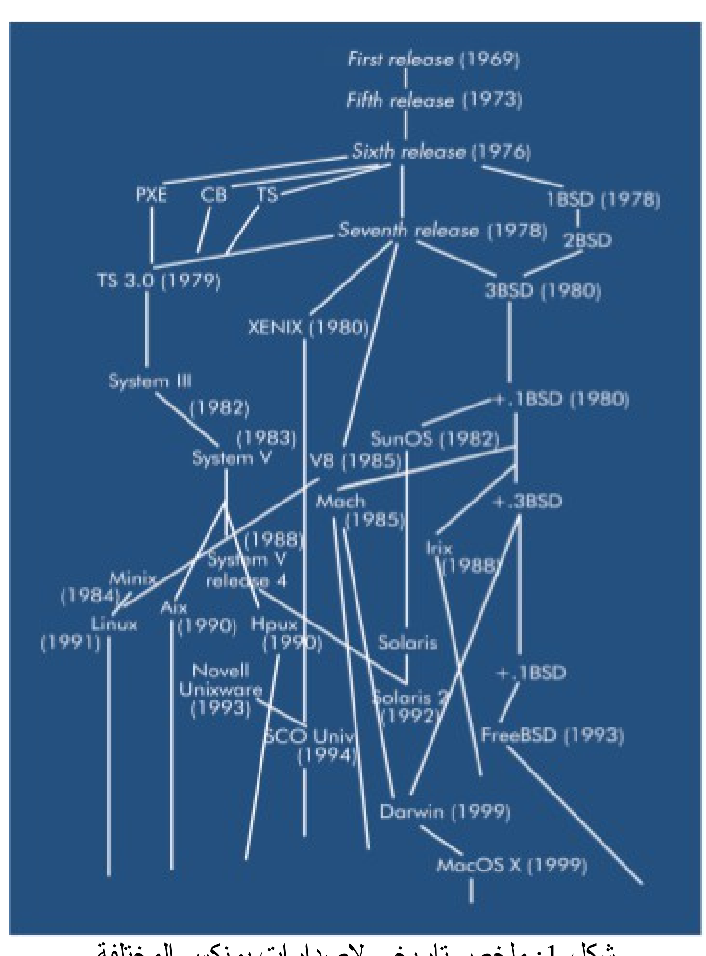
لقد تفرّع نظام يونكس إلى فروع عديدة مع الوقت، كان أهمها System V من AT&T، و BSD من جامعة كاليفورنيا. معظم أنظمة يونكس الحاليّة مبنيّة على أحدهما أو مزيج من الاثنين.

لكن كانت AT&T في ذلك الوقت تحت طائلة قضايا قانونيّة تتعلق باحتكار الهاتف (حيث كانت شبكة الهاتف الرائدة – إن لم تكن الوحيدة – في الولايات المتحدة الأمريكيّة)، مما أجبرها على الانقسام إلى عدد من الشركات الأصغر، مما أدى بحقوق يونكس بالترنُّح بين الملّاك: ففي عام 1990 كانت الحقوق مشتركة مناصفة بين Open Software Foundation

---

3 ويشار إليه أحياناً بـ SysV

OSF – و UI – UNIX International، التي سميت لاحقاً UNIX Systems Laboratory – USL، والتي قاضت جامعة بيركلي على نسخ BSD التي لديها، لكنها خسرت، حيث لم تتضمّن الرخصة الأصليّة أيّة حقوق ملكية متعلقة بأكواد يونكس. بعد ذلك بيعت حقوق يونكس لشركة نوفل التي تنازلت عن جزء منها إلى SCO، والآن لا يبدو واضحاً من يملك هذه الأجزاء، حيث طالبت كل من نوفل وOSF و SCO بحقوق ملكيتها في عدد من المواجهات. ومن الأمثلة الحديثة على هذه المشكلة قضية SCO، والتي رفعت قضية على شركة IBM وذلك لأنها – حسب أقوال SCO – أضافت أجزاء من المصدر البرمجي ليونكس إلى بعض إصدارات نواة لينكس، حيث زعمت SCO أن هذه الأكواد حَوَت بعض أكواد يونكس الأصليّة. النتيجة هذه الأيام هي بقاء القضية في المحاكم، حيث تحولت SCO إلى وحش صناعة التكنولوجيات الذي يهدد لينكس و IBM و مستخدمي أنظمة يونكس المملوكة الأخرى، وذلك بتأكيدها على الحقوق الأصليّة ليونكس وأن على الجميع أن يدفعوا لهم. علينا أن نرى كيف تتطور هذه القضية، وكذلك كيف تتطور قضية ملكية يونكس معها.



شكل 1: ملخص تاريخي لإصدارات يونكس المختلفة

السيناريو الحالي ليونكس تغير كثيراً منذ ظهور لينكس عام 1991، حيث صار هذا النظام (بين عامي 1995 و

1999) بديلاً حقيقيًّا لأنظمة يونكس المملوكة، وذلك نتيجة لكثرة منصات العتاد التي يدعمها، والدعم القوي للمجتمع العالمي والشركات التي تساهم في تقدّمه. يستمر عدد من الإصدارات المملوكة ليونكس بالبقاء في السوق، وذلك بسبب تطويعها للبيئات الصناعيّة وذلك لكونها أفضل نظام تشغيل في السوق، ولأن هناك احتياجات لا يمكن تلبيتها إلا باستخدام يونكس وأنظمة عتاد معيّنة.[4] إضافة إلى ذلك، فبعض أنظمة يونكس المملوكة أفضل من جنو/لينكس في كلّ من الموثوقيّة والأداء، رغم أن الهوة تتضاءل تدريجيًّا مع الوقت، حيث تبدي الشركات التي لديها أنظمة يونكس خاصة بها اهتماماً متزايداً بأنظمة جنو/لينكس، وتوفر بعضاً من تطويراتهم لإضافتها إلى لينكس. يمكننا أن نتوقع ما يشبه الانقراض لأنظمة يونكس المملوكة مع الوقت، ليتم إحلالها بتوزيعات مبنية على لينكس يتم تطويرها على أيدي الشركات المصنّعة للعتاد لتتوافق مع معدّاتهم.

استعراض لهذه الشركات:

1) SUN[5]: توفر نظام يونكس اسمه سولاريس (طورته عن SunOS). بدأ سولاريس كنظام BSD، لكن أغلبه الآن System V مع أجزاء من BSD؛ استخدامه الحاليّ في الغالب على أجهزة SPARC من شركة Sun، وفي الأجهزة ذات المعالجات المتعدّدة وحتّى 64 معالجاً. تروج Sun لأنظمة جنو/لينكس كبيئة تطوير جافا، ولديهم توزيعة لينكس تعرف بنظام جافا المكتبي والمنتشرة بكثرة في عدد من الدول. لقد بدأت أيضاً باستخدام بيئة سطح المكتب جنوم، وتوفر دعماً ماليًّا لعدد من المشاريع، ومنها: موزيلا و جنوم والمكتب المفتوح. ربما علينا أن نذكر أيضاً مبادرتها في الإصدار الأخير ليونكس سولاريس، حيث حرّرت معظم أكواده في الإصدار العاشر منه. إنشاؤها لمجتمعات لكل من معماريتي إنتل و سبارك، سمّيت OpenSolaris، جعل من إنشاء توزيعات حرّة لسولاريس ممكناً.[6] ربما يجدر بنا أيضاً ذكر المبادرات التي قامت بها عام 2006 لتحرير منصة جافا ضمن تراخيص GPL، كمشروع OpenJDK.

---

4 لا تنس عزيزي القارئ أن الكاتب يتحدث هنا عن المدة الواقعة بين 1991 و 1999، وليس الوقت الذي تقرأ به هذا الكتاب.

5 تعاود حاليًّا كلّ أملاك Sun إلى شركة Oracle التي استحوذت عليها.

6 يوجد الآن مشروع اسمه OpenIndiana، وهو خلف OpenSolaris بعد أن استحوذت أوراكل على Sun.

2) BSD: رغم أن BSD ليس شركة كالمذكورة آنفاً، إلا أن تطويره ما زال مستمراً، مع وجود مشاريع متفرعة عنه، كمشروع FreeBSD، و NetBSD، و OpenBSD (الذي يعتبر آمَن أنظمة يونكس)، و trustedBSD وغيرها. ستوفر أنظمة التشغيل هذه تحسينات أو تعاوناً في البرمجيات مع جنو/لينكس أيضاً، عاجلاً أو آجلاً. إضافة إلى هذا، لدينا نواة داروين والتي تتفرع من BSD 4.4، والتي طورتها Apple كنواة مفتوحة المصدر لنظام لتشغيل MacOS X.

3) مايكروسوفت: بعيداً عن عرقلتها تطوير يونكس وجنو/لينكس بالتسبب بعدم توافقية بين التقنيات، فليس لها أيّة مشاركة مباشرة في أنظمة جنو/لينكس.[7] طورت مايكروسوفت في بداياتها (عام 1980) نظام Xenix للأجهزة الشخصيّة، مبنيًّا على ترخيص يونكس من AT&T، والذي رغم أنه لم يُبَع مباشرة، فقد بيع عبر وسطاء مثل SCO، والتي استحوذت على Xenix عام 1987 لتعيد تسميته إلى SCO UNIX عام 1989. مما يثير الفضول أن مايكروسوفت اشترت حقوق يونكس من SCO، والتي كانت اشترتها من نوفِل. دوافع مايكروسوفت لهذا الاستحواذ ليست واضحة، لكن البعض يعتقدون بوجود علاقة بين هذه الخطوة وكون مايكروسوفت تساند SCO في دعوى قضائيّة ضد IBM. لقد قامت مايكروسوفت حديثاً، وتحديداً عام 2006، بالوصول إلى اتفاق مع نوفل (والتي توفر حالياً توزيعة SuSE ومجتمع OpenSuse)، بعدد من القرارات ثنائيّة الجانب، وذلك لعمل تسويق لكلتي المنصتين. رغم هذا، فمجتمع لينكس لا يزال مرتاباً من إمكانية وجود معنى ضمني متعلق بالملكية الفكرية لنظام جنو/لينكس، وقضايا قد تسبب مشاكل قانونيّة بما يتعلق باستخدام براءات الاختراع.

ومن الطرائف التاريخيّة الملفتة الأخرى، أنها أطلقت بالتعاون مع شركة اسمها UniSys حملة تسويقية عن كيفية التحويل من أنظمة يونكس إلى أنظمة ويندوز، وعلى الرغم من أن الهدف ربما لم يكن حميداً، فمن الحقائق الملفتة أن خادم الوِب المستخدم لهذا العمل كان خادم أباتشي على FreeBSD. ومِن آنٍ لآخَر، تدفع مايكروسوفت لشركات "مستقلة" (قد يقول البعض بأنها ليست مستقلة كفاية) لعمل تحليل لمقارنة الأداء بين وندوز من جهة، ولينكس ويونكس من جهة أخرى.

---

7 لقد تغير هذا الوضع مؤخراً، حيث ساهمت بإضافة تعاريف أجهزتها التخيلية VirtualPC إلى نظام لينكس، إضافة لتوفير منصة DotNet.

وبخلاصة عامّة، فإن بعض التعليقات التي تظهر في الكتب المتعلقة بيونكس تشير إلى حقيقة كون يونكس نظاماً بسيطاً ومترابطاً بشكل منطقي من الناحية التقنيّة، ومصمّم بناءً على أفكار جيدة وضعت حيّز التنفيذ، لكن علينا ان لا ننسى في المقابل أن بعض هذه الأفكار تم الحصول عليها من الدعم الذي توفره مجتمعات المستخدمين والمطورين بتحمّس، حيث يتعاونون على تطوير التقنية ودعم تطورها.

وبما أن التاريخ يعيد نفسه على ما يبدو، فذاك التطور والحماس مستمر مع أنظمة جنو/لينكس.

3 أنظمة جنو/لينكس

لم يكن لدى مستخدمي الحواسيب الشخصية الأولى قبل عشرين عاماً خيارات عديدة لأنظمة التشغيل. لقد كان نظام Microsoft DOS يسيطر على سوق الحواسيب الشخصيّة. بينما كان نظام ماك من شركة أبل من الخيارات، لكن ثمنه كان باهظاً مقارنة ببقيّة الخيارات. وقد كان يونكس من الخيارات الأخرى التي كانت متوفّرة للأجهزة الضخمة والمكلفة.

كان Minix من أول الخيارات ظهوراً - عام 1984 – والذي طوره أندرو تانيبوم من الصفر لاستخدامه لأغراض تعليمية، حيث أراد أن يستخدمه لتعليم طريقة تصميم وتنفيذ أنظمة التشغيل.

لقد كان التصور الذي بني عليه نظام Minix هو العمل على أجهزة Intel 8086 التي كانت منتشرة بكثرة في ذلك الوقت، حيث كانت الأساس للحواسيب الشخصيّة من IBM. الميزة الأساسيّة لهذا النظام برزت من مصدره البرمجي الذي كان متاحاً للجميع (إثنا عشر ألف سطر برمجيّ بلغة التجميع ولغة C)، ومتوفرة في كتب تانيبوم التعليميّة المختصة بأنظمة التشغيل. لكنه كان أداة تعليميّة أكثر مما كان نظاماً كفؤًا مصمماً للأداء والأنشطة العمليّة.

وفي التسعينات، شجّعت مؤسسة البرمجيات الحرّة FSF – عبر مشروع GNU التابع لها – العديد من المبرمجين للتشجيع على إنتاج برمجيات جيّدة وتوزيعها بحرّيّة. وبجانب العمل على برمجة الأدوات، فقد كان المشروع يعمل على برمجة نواة نظام تشغيل تدعى HURD، والتي كانت بحاجة لعدّة سنوات لتصير جاهزة.

وفي ذلك الوقت، وتحديداً في تشرين أول/أكتوبر عام 1991، عرض طالب فنلندي اسمه لينوس تورفالدز الإصدار 0.0.1 من نواة نظام تشغيل كتبه بنفسه، وسمّاها لينكس. كان لينكس مصمّماً لأجهزة Intel 386، وقدمه لينوس إلى مجتمعات المبرمجين ومجتمع الإنترنت تحت رخصة GPL لاختباره، ولمساعدته في التطوير إذا أعجبهم. لقد كان هناك حماس شديد بين من رأوه، حيث ما إن رأوه حتى أصبح هناك عدد كبير من المبرمجين يعملون على تطوير النواة وإنتاج برامج لها.

من المزايا التي تميّز بها لينكس عن أنظمة التشغيل الأخرى في ذلك الوقت، والتي وُرِث بعضها من يونكس، والتي ما زالت موجودة حتى لآن:

1) لينكس نظام تشغيل مفتوح المصدر: يمكن لأيٍّ كان الوصول إلى مصادره البرمجية، وتعديلها، وعمل إصدارات جديدة يمكن نشرها تحت رخصة GPL، مما يجعله نظاماً حرًّا بالفعل.

2) المحموليّة (قابلية النقل): لينكس (كنظام يونكس الأصليّ) مصمم كي لا يعتمد كثيراً على معمارية جهاز معيّن؛ نتيجة لذلك، نظام لينكس مستقل عن الجهاز الهدف الذي سيعمل عليه، مما يجعل نقله إلى أيّ معماريّة لها مُصرِّف C، كمُصرِّف GCC من GNU ممكناً. هناك أجزاء بسيطة بلغة التجميع، ومعرفات لعدد بسيط من الأجهزة التي تعتمد على جهاز معيّن، والتي يتوجّب إعادة كتابتها عند النقل لأيّة معماريّة جديدة. بفضل هذا، نظام جنو/لينكس من أنظمة التشغيل التي تعمل على أكبر عدد من المعماريات: x86 و IA86 من إنتل، و x86 و x86_64 من AMD، وسبارك من Sun، و MIPS من Silicon، و powerPC الذي استخدمته أبِل، و IBM S390، وألفا من كومباك، و m68k من موتورولا، و Vax، و ARM، و HPPArisc ...

3) نواة أحاديّة Monolithic: التصميم الأساسيّ للنواة يجمع كلّ ما هو أساسيّ في قطعة واحدة، ولكنها مجزّأة تصوّريًّا في مهامها المختلفة. من المدارس الأخرى في أنظمة التشغيل، مدرسة الأنوية المجزّأة أو المصغّرة Microkernel (مثل Mach)، والتي تعمل فيها الخدمات كعمليات منفصلة، تتخاطب فيما بينها عبر نواة بسيطة (مصغّرة). بنيت نواة لينكس على التصوّر الأحاديّ، لأنه يصعب الحصول على أداء جيد من نموذج الأنوية المصغّرة (وذلك لصعوبة هذا الأمر وتعقيده). وفي نفس الوقت، مشكلة الأنوية الأحاديّة هي أنها عندما تكبر تصير ضخمة جدًّا، ويصير تطويرها صعباً؛ استُخدِم التحميل المتغيّر للأجزاء (modules) لحلّ هذه المشكلة.

4) الوحدات (modules) التي يتم تحميلها عند الحاجة: هذا يجعل وجود أجزاء من نظام التشغيل - كأنظمة الملفات ومعرّفات الأجهزة - كأجزاء خارجية يتم تحميلها (أو ربطها) مع النواة أثناء عملها عند الحاجة لذلك أمراً ممكناً. هذا يجعل من الممكن تبسيط النواة وتوفير هذه الأجزاء كعناصر يمكن تطويرها بشكل منفصل. مع استخدام هذه الوحدات، يمكن اعتبار نواة لينكس نواة مختلطة[8]؛ وذلك لأنها أحاديّة، ولكنها توفر عددًا من الوحدات المكمّلة

---

8 مترجم هذا الكتاب يعتبرها نواة هجينة hybrid بوجود هذه الأجزاء، مما يجعلها تحوي مزايا كلتي البنيتين الأحاديّة والمصغّرة؛ إلا أنّ المزايا الأساسيّة يمكن توفيرها في ملف وواحد، مما يعني أنها تبقى في الأصل نواة أحاديّة.

للنواة (مثلها في ذلك مثل الأنوية المصغّرة).

5) نظام يطوّره مجتمع تربطه الشبكة العالمية (الإنترنت) : لم تطور أنظمة تشغيل يوماً بهذا التوزع والامتداد، حيث لا يخرج تطوير الأنظمة عادة عن نطاق الشركة التي تطوّرها (في حالة الأنظمة المملوكة) أو المجموعة الصغيرة للمؤسسات الأكاديميّة التي تتعاون لإنشاء نظام معيّن. ظاهرة مجتمع لينكس تسمح للجميع بالمشاركة بما يسمح به وقت ومعرفة هذا الشخص. والنتيجة هي وجود مئات أو آلاف من مطوري لينكس. إضافة إلى هذا، وبسبب كون لينكس مفتوح المصدر بطبيعته، فهذا يجعله مثاليًّا للمختبرات لاختبار أفكار لأنظمة التشغيل بأقلّ تكلفة، حيث يمكن تنفيذ تلك الأفكار واختبارها ويمكن أخذ قياسات وإضافة أفكار إلى النواة في حال نجاحها.

أنجحت هذه المشاريع بعضها بعضاً، فبعيداً عن النواة، انضمّ إلى فريق FSF وبرمجيات أدوات جنو والأهم بينها مُصرِّف لغة سي GCC مشاريع أخرى مثل Xfree (إصدار الأجهزة الشخصيّة لـ X Window) ومشاريع أسطح مكتب مثل KDE و Gnome. وتطوُّر الإنترنت كنتيجة لمشاريع مثل خادم الوِب أباتشي، ومتصفّح موزيلا، وقواعد بيانات MySQL و PostgreSQL أدت إلى توفير تغطية جيّدة من التطبيقات لنواة لينكس لبناء أنظمة جنو/لينكس والمنافسة على مستوى يضاهي الأنظمة المملوكة، و إلى تحويل أنظمة جنو/لينكس إلى مثال للبرمجيات مفتوحة المصدر. صارت أنظمة جنو/لينكس رأس حربة مجتمع المصدر المفتوح، كنتيجة لعدد المشاريع التي تمكّنوا من ربطها للتكامل معاً بنجاح.

ساعدت نشأت شركات جديدة أنشأت بدورها توزيعات جنو/لينكس (تحزيم النواة والتطبيقات) ودعمتها، كأمثال ردهات وماندريك وسوزي في تقديم جنو/لينكس إلى الشركات المترددة وبدء النمو المطّرد الذي نشهده اليوم.

سنعلّق أيضاً على المناظرة المتعلّقة بإعادة تسمية الأنظمة إلى جنو/لينكس. تستخدم كلمة لينكس للإشارة إلى نظام التشغيل هذا كنوع من التبسيط، لكن هذا بنظر البعض يستخفّ بالعمل الذي قامت به FSF ومشروع جنو، والتي وفّرت الأدوات الأساسيّة للنظام. رغم ذلك، فمصطلح لينكس مستخدم بشكل واسع على النطاق التجاري للإشارة إلى نظام التشغيل بأكمله.

بشكل عام، المصطلح الأنسب والذي يعكس مساهمات المجتمع هو لينكس عندما يشار إلى نواة نظام التشغيل فقط.

لقد تسبب هذا ببعض الالتباس وذلك لأن الناس يتحدثون عن نظام التشغيل لينكس للاختصار. عندما نتعامل مع نظام التشغيل جنو/لينكس فنحن نتعامل مع سلسلة من الأدوات التي هي بالعادة مشروع جنو على نواة لينكس. ولهذا، فالنظام بالأساس جنو مع نواة لينكس. كان الهدف الأساسيّ لمؤسسة البرمجيات الحرّة من مشروع جنو إنشاء نظام تشغيل مفتوح المصدر على نمط يونكس وتسميته جنو.

قرر لينوس تورفالدز عام 1991 ضمّ نواة لينكس إلى أدوات جنو حيث لم يكن لدى مؤسسة البرمجيات الحرة نواة نظام تشغيل بعد. نواة جنو اسمها هيرد HURD، وهناك الكثير من العمل عليها الآن، وهناك إصدارات تجريبيّة متاحة لتوزيعات جنو/هيرد (راجع باب "إدارة النواة" للمزيد).

يقدّر أنه في توزيعة جنو/لينكس عادية، هناك 28% من أكواد جنو 3% تعود للنواة، والباقي يعود لأطراف أخرى كتطبيقات أو أدوات.

للوقوف على مساهمات جنو، يمكننا إلقاء نظرة على مساهماتها المضمّنة في أنظمة جنو/لينكس:

- مُصرِّف C و ++C
- صَدَفة Bash
- محرر Emacs (أو GNU Emacs).
- مفسّر postscript المسمّى ghostscript.
- مكتبة C القياسيّة، المعروفة بـ GNU C Library أو glibc.
- المُنقِّح المسمّى gdb
- makefile أو GNU make
- المجمّع GNU assembler المسمّى gas
- GNU Linker المسمّى gld

ليست أنظمة جنو/لينكس الوحيدة التي تستخدم برمجيات جنو، فمثلاً، أنظمة BSD أيضاً تعتمد على أدوات جنو. وبعض الأنظمة المملوكة، كنظام MacOS X من شركة أبِل يستخدم برمجيات جنو أيضاً. لقد أنتج مشروع جنو برمجيات ذات جودة عالية تم استخدامها في معظم توزيعات الأنظمة المبنية على يونكس الحرّة المملوكة.

من العدل أن يعرف العالم عمل الجميع وذلك بتسمية الأنظمة التي سنتعامل معها جنو/لينكس.

4 صورة عامّة عن مدير الأنظمة

تعتمد الشركات والمؤسسات الكبيرة أكثر فأكثر على موارد تكنولوجيا المعلومات لديها وطريقة إدارتها وتطويعها للقيام بالمهام المطلوبة منها. الازدياد المطّرد للشبكات الموزّعة وأجهزة الخوادم والعملاء ولّدت طلباً كبيراًعلى وظيفة جديدة في سوق العمل: وهي ما يطلق عليه مدير النظام أو مدير الأنظمة.

مدير الأنظمة مسؤول عن عدد كبير من المهام الهامّة. يكون لدى أفضل مدراء الأنظمة معرفة عامّة جيّدة في الجانب النظري والعملي. يمكنهم القيام بمهام مثل: تركيب الأسلاك وإصلاحها، وتثبيت أنظمة التشغيل أو التطبيقات البرمجية، وإصلاح مشاكل وأخطاء الأنظمة عتادية كانت أو برمجيّة، وتدريب المستخدمين، وتقديم خدع وتقنيّات لتحسين الإنتاجية في مجالات تتراوح بين معالجة النصوص وحتى التصميم بمساعدة الحاسوب CAD وأنظمة المحاكاة المعقّدة، وتخمين تكلفة شراء المعدّات العتادية والبرمجية، وأتمتة عدد كبير من المهام المشتركة، وزيادة الأداء العام للمؤسسة.

يمكن اعتبار مدير الأنظمة الموظف الذي يساعد المؤسسة على الاستفادة بأكبر قدر ممكن من الموارد المتاحة، وذلك لتتمكن المؤسسة بأكملها من التحسُّن.

يمكن إنشاء العلاقة مع المستخدم النهائي للمؤسسة بأكثر من طريقة: إما عن طريق تدريب المستخدمين، أو بتوفير المساعدة المباشرة عند وجود مشاكل. مدير الأنظمة هو الشخص المسؤول عن ضمان أن التقنيات التي يستخدمها المستخدمون تعمل بالشكل المطلوب، مما يعني بأن الأنظمة تفي بتطلّعات المستخدمين والمهام التي يحتاجون للقيام بها.

لم يكن لدى العديد من الشركات والمؤسسات قديماً – بل وحتى الآن – رؤية واضحة لدور مدير النظام. عندما كانت حوسبة الأعمال في أيامها الأولى (في الثمانينات والتسعينات)، كان يُرى مدير النظام على أنه الشخص الذي يفهم الحاسوب والمسؤول عن تركيب الأجهزة ومراقبتها وإصلاحها في حال وجود أيّة مشكلة. كانت الوظيفة في العادة قائمة على فنّي حاسوب متعدّد المهام مسؤول عن حلّ المشاكل متى وأينما ظهرت. لم يكن هناك صورة عامّة واضحة عن الوظيفة، لأن المعرفة المتقدمة لم تكن مطلوبة آنذاك، بل فقط معرفة بسيطة في مجموعة صغيرة من التطبيقات (والتي عادة ما تبقى في حدود عشرة أو اثني عشر تطبيقاً على الأكثر) من محررات النصوص وبرامج الجداول وقواعد البيانات وغيرها، ومعرفة بسيطة في العتاد، وهو ما كان

يهمّهم للتعامل مع المهام اليوميّة. ولهذا، فقد كان بإمكان أيّ شخص لديه هذه المعرفة البسيطة ويعلم المطلوب منه أن يقوم بهذه الوظيفة، مما يعني أن مدراء الأنظمة في العادة لم يكونوا تقنيي حاسوب بالمعنى المتعارف عليه، وقد كانت المعرفة في العادة تنقل بين المدير الحاليّ أو القديم و المتدرّب.

لقد عكس هذا الموقف إلى حدّ ما بداية تاريخ إدارة الأنظمة (رغم وجود أناس حتى الآن يعتقدون بأنها بالأساس نفس الوظيفة). أما الآن – في عصر الإنترنت والخوادم الموزّعة – مدير الأنظمة متخصّص (موظف بدوام كامل لهذا الغرض فقط) يوفر خدمات في مجال عتاد وبرمجيات الأنظمة. على مدير الأنظمة تنفيذ عدّة مهام على عدّة أنظمة تقنيّة معلومات، هي في العادة خليط من الأنظمة من أنواع مختلفة، وذلك لجعلها صالحة لعدد من المهام.

يحتاج مدراء الأنظمة هذه الأيام معرفة عامّة نظرية وعملية في عدد من المجالات، من تقنيات الشبكات، إلى أنظمة التشغيل، والتطبيقات المختلفة، ومعرفة أساسيّة بالبرمجة بعدد كبير من اللغات، ومعرفة موسّعة في العتاد – بما يتعلّق بالحاسوب نفسه وبالملحقات المرتبطة به –، وتقنيات الإنترنت، وتصميم المواقع، وإدارة قواعد البيانات. وعادة ما تكون النظرة العامة مرتبطة بمجال عمل الشركة كالكيمياء، الفيزياء، الرياضيات ...إلخ. ولهذا، لا تتفاجأ إذا علمت بأن أيّ شركة متوسطة أو كبيرة توقّفت عن توظيف التقني الذي تجده، وتحولت إلى توظيف مجموعة صغيرة من المتخصّصين ذوي المعرفة المعمّقة، وعادة من حاملي الشهادات الجامعيّة، ولكلّ منهم مهام معيّنة داخل المؤسسة.

> على مدير الأنظمة أن يكون قادراً على إتقان مدى واسع من التقنيات وذلك ليتمكن من التكيف مع المهام المختلفة التي قد تظهر داخل المؤسسة.

بسبب كثرة المعلومات المطلوبة، ليس من الغريب وجود عدد من الأنواع الفرعيّة لمدراء الأنظمة. من العاديّ أن نرى في المؤسسات الكبيرة عددًا من مدراء أنظمة التشغيل (يونكس، ماك، وندوز): مدير قواعد بيانات، مدير نسخ احتياطيّة، مدير أمن معلومات، مدراء أنظمة مهمتهم مساعدة المستخدمين، ...إلخ.

في المؤسسات الصغيرة، من الممكن أن توكل كلّ أو بعض هذه المهام إلى مدير نظام واحد أو عدد قليل من مدراء الأنظمة. عادة ما يكون لمدراء أنظمة يونكس (أو جنو/لينكس) بعض هذه المهام (إلا إذا كان هناك مدير نظام مسؤول عن

كلّ المهام). عادة ما تكون المنصّة التي يعمل عليها مدير النظام هي أحد أشكال يونكس (وفي حالتنا جنو/لينكس) والذي يجعل هذه الوظيفة تتطلب قَدْراً كافياً من العناصر المحدّدة لتكون فريدة من نوعها. يونكس (بأشكاله المختلفة) نظام تشغيل مفتوح وقويّ جدًّا، وكأيّ نظام برمجيّ، يحتاج قدراً معيّناً من التطويع والضبط والصيانة والمتابعة للقيام بالمهام التي سيعمل على تلبيتها. ضبط وصيانة أنظمة التشغيل وظيفة حَرِجَة، وفي حالة يونكس، قد تصير محبطة نوعاً ما.

من القضايا الحرجة التي سيتم تغطيتها ما يلي:

1) إنّ حقيقة كون النظام قويًّا جدًّا تعني أيضًا أن هناك إمكانيّة كبيرة لتطويعه (أو ضبطه) للمهام التي نريد القيام بها. علينا تقييم الإمكانيّات التي يوفرها لنا والتي تناسب هدفنا النهائيّ.

2) من الأمثلة الواضحة على الأنظمة المفتوحة جنو/لينكس، والذي سيوفر لنا تحديثات دائمة، وذلك لإصلاح علل في النظام، أو لتضمين مزايا جديدة. ومن البديهيّ أن لكلّ هذا أثر مباشر واضح على تكلفة الصيانة التي هي جزء من المهام الإدارية.

3) يمكن استخدام الأنظمة للمهام ذات التكلفة الحرجة، أو في الأماكن الحرجة للمؤسسة، حيث لا يمكن السماح بتوقف واضح يحدّ من أو يؤثر على أداء المؤسسة.

4) الشبكات حاليًّا نقطة هامّة (إن لم تكن الأهم)، ولكنها أيضاً مكان حرج جدًّا للمشاكل، وذلك نتيجة لطبيعتها الموزعة، وتعقيد النظام في إيجاد وتنقيح وإصلاح المشاكل التي قد تظهر.

5) في حالة نظام يونكس – ونظام جنو/لينكس الذي سنعمل عليه – فالعدد الهائل من الإصدارات والتوزيعات المختلفة تزيد من مشاكل إدارتها، وذلك لأنه من المهم معرفة المشاكل والاختلافات في كلّ توزيعة.

مهام إدارة الأنظمة وإدارة الشبكات لها مزايا مختلفة، وعادة ما يتم التعامل مع كلّ منها على حدة (أو بمدراء مختلفين). رغم أنه من الممكن أن ننظر إليهما كوجهين لعملة واحدة، حيث النظام نفسه (عتادًا وبرمجيات) من جهة، والبيئة (الشبكة) التي تتعايش فيها الأنظمة معاً من جهة أخرى.

عادة ما تُفهَم إدارة الشبكات على أنها إدارة النظام كجزء من الشبكة، وتشير إلى الأجهزة والخدمات القريبة المطلوبة لعمل الجهاز في البيئة الشبكيّة، لا يشمل ذلك أجهزة الشبكة كالمُبَدِّلات switches، والجسور bridges، والموزعات المركزية hubs، وأجهزة الشبكة الأخرى، لكن المعرفة الأساسيّة ضروريّة لتسهيل المهام الإداريّة.

في هذا المساق، سنتعامل بداية مع الجوانب المحلية للنظام نفسه، ومن ثمّ سنبحث في مهام إدارة شبكة وخدماتها.

لقد ذكرنا سابقاً مشكلة صعوبة وضع تعريف محدّد لمدير النظام، وذلك لأن سوق تقنيّة المعلومات ليس واضحاً. كان من الشائع طلب مدراء أنظمة بناء على مجموعات (اختلقتها الشركات) من المبرمجين أو مهندسي البرمجيات لم تكن مناسبة بشكل كامل.

المبرمج عادة منتج أكواد، وفي هذه الحالة، لن يحتاج مدير النظام إلى إنتاج الكثير منها، وذلك لأنها قد تكون هامّة في مهام معينة، لكن ليس فيها كلّها. من المطلوب عادة من مدير النظام أن يكون لديه معرفة في مجالات معينة حسب نوع العمل، وذلك كالتالي:

1) مؤهِّل ما أو درجة جامعيّة، ويفضّل في تقنيّة المعلومات، أو في مجال ذي علاقة مباشرة بمجال عمل الشركة أو المؤسسة. تحوي الصورة النمطية عن مدير النظام مجال دراسة ذي علاقة بهندسة أو علم الحاسوب أو في مجال أنشطة المؤسسة مع خبرة موثّقة في المجال ومعرفة واسعة في الأنظمة الهجينة وتقنيات الشبكات.

2) من الاعتياديّ أن تُطلَب خبرة من سنة إلى ثلاث سنوات في إدارة الأنظمة (إلا إذا كانت الوظيفة كمساعد لمدير نظام موجود). قد يُطلب أيضاً خبرة من 3 إلى 5 سنوات.

3) معرفة أساسيّة او معمّقة في بيئات الشبكات وخدماتها. خدمات موافيق[9] TCP/IP، و ftp, telnet, ssh, http, nfs, nis, ldap ...إلخ.

9 موافيق: جمع ميفاق، يقابلها في الإنجليزيّة protocol

4) معرفة بلغات النصوص البرمجيّة[10] لعمل النماذج الأوليّة للأدوات أو لأتمتة المهام بسرعة، ومن الأمثلة عليها: shell scripts و Perl و TCL و Python وغيرها، وخبرة في البرمجة بعدد كبير من اللغات، مثل C و ++C وجافا ولغة التجميع وغيرها.

5) قد تُطلَب أيضاً خبرة في تطوير تطبيقات ضخمة في أيٍّ من هذه اللغات.

6) معرفة معمَّقة بسوق تقنيّة المعلومات – في كلّ من العتاد والبرمجيات – في حال الحاجة لتقدير تكاليف الشراء، أو تركيب أنظمة جديدة، أو التجهيز الكامل من البداية.

7) خبرة في أكثر من إصدار واحد من يونكس (أو أنظمة جنو/لينكس)، مثل Solaris, AIX, AT&T System V, BSD وغيرها.

8) خبرة في أنظمة تشغيل غير يونكس، قد تكون هناك أنظمة إضافيّة في المؤسسة، مثل: وِندوز 9x/NT/2000/Vista/Se7en أو Mac OS أو VMS أو أنظمة IBM وغيرها.

9) معرفة متينه بتصميم وتنفيذ يونكس، وآليات ترحيل وتبادل البيانات، والاتصالات البينيّه للعمليات، والمتحكّمات، وغيرها. إذا كانت المهام الإدارية تتضمن تحسين النظام (ضبطه) على سبيل المثال.

10) معرفة وخبرة في أمن المعلومات: إنشاء جدران الحماية، وأنظمة الاستيثاق، وتطبيقات التشفير، وأمن أنظمة الملفات، أدوات مراقبة الأمن، وغيرها.

11) خبرة في قواعد البيانات، ومعرفة في SQL وغيرها.

12) تركيب وإصلاح العتاد و/أو أسلاك وأجهزة الشبكة.

---

10 لغات النصوص البرمجية أو script(ing) languages هي لغات برمجة تعتمد على مفسّر interpreter يعمل على قراءة وتنفيذ السطور البرمجية سطراً بسطر، مما يجعل الكود البرمجي والبرنامج التنفيذيّ واحداً، أي أنك لن تحتاج لتصريف المصدر البرمجي لكي تستخدمه.

5 مهام مدير النظام

كما شرحنا سابقاً، بإمكاننا تقسيم مهام مدير أنظمة جنو/لينكس (أو يونكس بشكل عام) إلى جزئي أساسيّين: مدير نظام ومدير شبكة. سنعرض في النقاط التالية بإيجاز ما تتكون منه المهام بشكل عام لأنظمة جنو/لينكس (أو يونكس)؛ معظم أجزاء محتوى هذا الدليل التعليميّ سيتم التعامل معها بقدْرٍ معيّن من التفاصيل؛ معظم هذه المهام الإداريّة سيتم تطويرها في هذا الدليل التعليميّ؛ لكن سيتم شرح أجزاء أخرى من هذه المهام بشكل سطحيّ لأسباب تتعلّق بالمساحة والتعقيد.

تشمل المهام الإداريّة سلسلة من التقنيات والمعارف، وما هذا الكتّاب إلا غيض من فيض؛ على أيّ حال، تشكّل المصادر الملحقة بكلّ وحدة مراجع للتوسّع في هذه المواضيع. وكما سنرى، هناك مراجع كثيرة لكلّ نقطة تم التعامل معها تقريباً.

يمكن تقسيم مهام إدارة النظام بشكل مبسط إلى قسمين، الأول إدارة النظام المحلي، والثاني إدارة الشبكة.

مهام إدارة النظام المحلي (دون ترتيب معيّن):

1) تشغيل وإيقاف النظام: أيّ نظام مبني على يونكس [أو متوافق معه] له أنظمة يمكن ضبطها لتشغيله وإيقافه، حيث يمكننا ضبط أيّ الخدمات تكون متوفرة عند تشغيل الجهاز، ومتى نحتاج لإيقافها، مما يمكننا من برمجة النظام لينطفئ للصيانة.

2) إدارة المستخدمين والمجموعات: إتاحة المجال للمستخدمين من المهام الأساسيّة لأيّ مدير نظام. سيكون علينا تحديد أي المستخدمين سيكونون قادرين على الوصول إلى النظام، وكيفية وصولهم إليه، والصلاحيات الممنوحة لهم؛ وأن يُنشئ مجتمعات باستخدام المجموعات. وهناك حالة خاصة، وهي مستخدمو النظام، وهم مستخدمون افتراضيون موجهون للقيام بمهام النظام.

3) إدارة موارد النظام: ما الموارد التي نوفرها؟ وكيف؟ ولِمَن نتيح الوصول إليها؟

4) إدارة نظام الملفات: قد يكون في الحاسوب وسائط مختلفة لتخزين البيانات، وأجهزة (أقراص صلبة وضوئيّة وثابتة، وغيرها) بأنظمة مختلفة للوصول إلى الملفات. حيث قد تكون هذه البيانات دائمة أو قابلة للإزالة او مؤقتة، مما يعني الحاجة لإنشاء وإدارة عملية تثبيت وإزالة تثبيت أنظمة الملفات التي تتيحها الأقراص والأجهزة المرتبطة بالنظام.

5) حصص النظام: سيكون على مدير النظام إدارة أيّة موارد مشتركة، وسيكون عليه إنشاء نظام حصص (اعتماداً على تعداد المستخدمين) لتفادي استخدام أحد المستخدمين الموارد بشكل كبير وغير مبرر، أو لتحديد تصنيفات (أو مجموعات) المستخدمين بناء على الاستخدام الأعلى أو الأقل لهذه الموارد. ومن المعتاد ضبط أنظمة الحصص لمساحة القرص أو الطباعة أو المعالج.

6) أمن النظام: يتعلق الأمن المحلي للنظام بحماية الموارد من الاستخدام غير المبرر أو الوصول غير المصرح به إلى بيانات النظام أو إلى بيانات المستخدمين الآخرين أو المجموعات الأخرى.

7) النسخ الاحتياطي واستعادة النظام: اعتماداً على أهمية البيانات، يحتاج مدير النظام لإنشاء سياسات مرتبطة بجدول زمني لعمل نسخ احتياطية للأنظمة لديه. سيكون عليه تحديد أوقات النسخ الاحتياطيّ، وذلك لحماية البيانات من انهيارات النظام أو العوامل الخارجية والتي يمكن أن تسبب تلف البيانات أو فقدانها.

8) أتمتة الأعمال المتكررة: يمكن أتمتة العديد من المهام الإدارية أو المهام المرتبطة بالاستخدام اليومي للجهاز بسهولة، وذلك لسهولتها وسهولة تكرارها، إضافة إلى توقيتها، مما يعني وجوب تكرارها على نمط معين. تتم هذه الأتمتة إما بالبرمجة بلغة مفسّرة (نصوص برمجية) بلغات مثل لغة الصدفة أو بيرل [أو بايثون] أو غيرها، أو بتضمين هذه المهام في أنظمة الجدولة، مثل crontab و at وغيرها.

9) إدارة الطوابير والطباعة: يمكن استخدام أنظمة يونكس كأنظمة طباعة لإدارة طابعة واحدة أو أكثر مرتبطة بالنظام، إضافة لإدارة طوابير العمل التي يمكن أن يرسلها المستخدمون أو التطبيقات إليها.

10) إدارة أجهزة المودِم (modem) والطرفيات: هذه الأجهزة شائعة في البيئات غير المتصلة بشبكة محلية أو إحدى شبكات النطاق العريض:

- تتيح المودِمات (modems) إمكانيّة الاتصال بشبكة عبر وسيط (مزود خدمة إنترنت ISP أو موفر وصول)، أو إمكانية الوصول للشبكة من الخارج، وذلك عبر اتصال هاتفي من أيّ نقطة في شبكة الهاتف.

- أما بالنسبة للطرفيات، فقد كان من المعتاد قبل ظهور الشبكات أن تكون أجهزة يونكس وحدات حوسبة مركزية، مع مجموعة من الطرفيات البدائيّة والتي كانت تستخدم فقط لعرض المعلومات أو السماح بإدخالها عبر لوحات مفاتيح خارجية؛ لقد كانت في العادة سلسلة من الطرفيات المرتبطة على التوازي. أما الآن، فهذه الطرفيات مازالت شائعة في البيئات الصناعيّة، كما ويوفر نظام جنو/لينكس المكتبي الذي يهمنا ميزة خاصة، وهي الوصول إلى الطرفية النصيّة الوهمية باستخدام مفتاحي Ctrl و Alt مع أحد أزرار المهام F1 – F12.

11) سجلّ النظام log: نحتاج لتطبيق سياسات تسجيل تُعْلِمُنا بفشل النظام أو أداء برنامج أو خدمة أو مورد معين عند الحاجة، وذلك لنتمكن من معرفة إذا كان النظام يعمل على الوجه الصحيح. أو لعمل ملخصات بالموارد المستخدمة واستخدامات النظام والإنتاجية على شكل تقرير.

12) ضبط أداء النظام: تقنيات ضبط النظام لأهداف معيّنة. عادة ما يكون النظام مصمماً للقيام بوظيفة معينة، ويمكننا التأكد من أنه يعمل على الوجه الصحيح (باستخدام التقارير مثلاً)، وذلك لإعادة النظر في إعداداته وضبطها لأداء الخدمة المطلوبة.

13) تفصيل النظام: إعادة ضبط النواة. في أنظمة جنو/لينكس مثلاً، الأنوية قابلة للتخصيص بشكل كبير، وذلك اعتماداً على المزايا التي نريد تضمينها، أو الأجهزة التي لدينا أو التي نأمل بإضافتها مستقبلاً، إضافة إلى المتغيرات التي تؤثر على أداء النظام أو التي تحصل عليها التطبيقات.

مهام إدارة الشبكة

1) واجهة الشبكة والاتصال: نوع واجهة الشبكة التي نستخدمها، سواء كانت للاتصال بشبكة محلية، أو شبكة أكبر، أو شبكة عريضة النطاق عبر تقنيات DSL أو ISDN [أو 3G]. وكذلك، نوع الاتصال الذي سيكون لدينا، على شكل خدمات أو طلبات.

2) توجيه البيانات: البيانات التي سترسل – من أين وإلى أين – اعتماداً على أجهزة الشبكة المتاحة، ووظيفة الجهاز ضمن الشبكة؛ قد يكون من الضروري إعادة توجيه مسار البيانات من أو إلى مكان واحد أو أكثر.

3) أمن الشبكة: الشبكة مصدر محتمل للهجمات، خاصة إذا كانت مفتوحة لأيّ نقطة خارجيّة (كالإنترنت)، ولهذا فقد تهدّد أمن الأنظمة أو بيانات المستخدمين. علينا حماية أنفسنا، واكتشاف ومنع الهجمات المحتملة بسياسة أمنية واضحة وفعّالة.

4) خدمات الأسماء: على الشبكة عدد هائل من الموارد المتاحة. تسمح لنا خدمات الأسماء بتسمية الكيانات (كالأجهزة والخدمات) لنتمكن من إيجاد مكانها. بوجود خدمات مثل DNS و DHCP و LDAP وغيرها، سيكون بإمكاننا تحديد أماكن الخدمات والأجهزة لاحقاً ...

5) خدمة معلومات الشبكة NIS: تحتاج المؤسسات الكبيرة لآليات لتنظيم الموارد والوصول إليها بفاعليّة. تكون نماذج يونكس القياسية – كولوج المستخدمين اعتماداً على كلمات مرور محلية – فعّالة عندما تكون الأجهزة والمستخدمون قلّة، ولكن عندما تكون لدينا مؤسسات كبيرة، بهيكليات شجرية، ومستخدمون يمكنهم الوصول إلى موارد عديدة بنمط موحّد أو كلًّا على حدة وبصلاحيات مختلفة وما إلى ذلك، فمن الواضح أن الطرق البسيطة المتبعة في يونكس ليست فعالة أو حتى ممكنة. ولهذا نحتاج أنظمة فعالة أكثر، وذلك للتحكم بكل هذه الهيكليات. تساعدنا الخدمات كأمثال NIS و +NIS و LDAP لترتيب هذه التعقيدات بطريقة فعالة.

6) نظام ملفات الشبكة NFS: في هيكليات أنظمة الشبكة عادة ما يحتاج بعض أو كلّ المستخدمين إلى

مشاركة المعلومات (مثل الملفات). أو ببساطة، الوصول إلى الملفات مطلوب من أيّ نقطة في الشبكة وذلك بسبب التوزيع المكاني للمستخدمين. توفر لنا أنظمة ملفات الشبكة (مثل NFS) وصولاً مباشراً للملفات، بغض النظر عن مكاننا على الشبكة.

7) أوامر يونكس البعيدة: ليونكس أوامر شبكة مباشرة، حيث يمكن تنفيذ أوامر تنقل المعلومات عبر الشبكة أو تسمح بالوصول إلى بعض خدمات الجهاز، بغض النظر عن الاتصال الفيزيائي. تحوي هذه الأوامر حرف r في بدايتها، والذي يعني remote، ومن الأمثلة عليها: rcp, rlogin, rsh, rexec، وغيرها، والتي تسمح بالقيام ببعض المهام عن بعد عبر الشبكة.[11]

8) تطبيقات الشبكة: تطبيقات الاتصال بخدمات الشبكة بنموذج يحوي تطبيقًا عميلاً يتصل بخدمة يوفرها جهاز آخر، مثل telnet (التي توفر صولاً تفاعلياً) و FTP (لنقل الملفات). أو أنه يمكننا خدمة أنفسنا بالخادم المناسب: خادم وصول عن بعد telnet، خادم نقل ملفات FTP، خادم web، ...إلخ.

9) الطباعة عن بعد: الوصول إلى خوادم الطباعة البعيدة، إما بالاتصال بخوادم الطباعة مباشرة، أو بأجهزة أخرى تتيح الطابعات المحلية الخاصة بها. يتيح هذا الطباعة المباشرة للمستخدمين والتطبيقات.

10) البريد الإلكتروني: من الخدمات الرئيسية التي توفرها أجهزة يونكس خادم البريد الإلكتروني، والذي بإمكانه تخزين رسائل البريد أو تمريرها إلى خوادم أخرى، إذا لم تكن موجّهة لمستخدميه. وفي حالة الوِب، يوفر نظام يونكس أيضاً منصة مثالية بخادم الوِب المناسب. ليونكس حصّة الأسد في سوق خوادم البريد الإلكتروني والوِب ، وهو أحد الأسواق الرئيسية له، والذي يملك فيه مكانة مرموقة. توفر أنظمة جنو/لينكس حلولاً مفتوحة المصدر للبريد الإلكتروني والوِب والتي تمثل أحد استخداماتها الأساسية.

---

11 الأوامر المشار إليها هنا والمبدوأة بحرف r ليست آمنة، وقد تم الاستغناء عنها في إصدارات جنو/لينكس الحديثة لصالح أوامر ssh المبدوءة بحرف s كما أشرنا إلى ذلك في باب إدارة الشبكة.

11) مدير النوافذ: يعتبر مدير النوافذ الرسومي X Window في جنو/لينكس (ومعظم أنظمة يونكس) نموذجاً خاصاً للترابط. يتيح هذا النظام تواصلاً مباشراً مع الشبكة، ويعمل بنموذج الخادم-العميل؛ يسمح هذا لتطبيقات أن لا تكون مرتبطة بالجزء المرئي منها والتفاعل مع ذلك الجزء عبر أجهزة الإدخال مطلقاً، مما يعني إمكانيّة وجود هذه التطبيقات في أيّ مكان على الشبكة. فعلى سبيل المثال، يمكننا تشغيل تطبيق معين على أحد أجهزة يونكس، بينما نرى النتائج الرسومية على شاشة جهاز آخر، ويمكننا إدخال البيانات عبر فأرة ولوحة مفاتيح عن بعد. إضافة إلى هذا، فالعميل – المسمى عميل X – مجرد مكون برمجي يمكن نقلة إلى أنظمة تشغيل أخرى، مما يعني إمكانية تشغيل تطبيقات على جهاز يونكس ورؤية واجهتها الرسوميّة على أيّ نظام آخر. إن ما يسمى طرفيات X حالة خاصة، فهي ببساطة طرفيات بدائية يمكنها إظهار أو التفاعل (باستخدام فأرة أو لوحة مفاتيح) مع تطبيق يعمل عن بعد فقط.

6 توزيعات جنو/لينكس:

عندما تحدثنا عن أصول جنو/لينكس، رأينا أنه لا يوجد نظام تشغيل وحيد ومحدد بشكل واضح. فمن ناحية، هناك ثلاثة عناصر برمجية يقوم عليها نظام جنو/لينكس، وهي:

1) نواة لينكس: كما رأينا، فالنواة هي الجزء المركزي من النظام فقط. ودون الأدوات والصدفات والمُصرِّفات والمحررات وغيرها، لم يكن يمكننا الحصول على نظام كامل.

2) تطبيقات جنو: أُكمِل تطوير لينكس عبر البرمجيات التي أوجدتها FSF عبر مشروع جنو، والتي وفرت محررات مثل emacs ومصنّفًا (gcc) وأدوات عديدة.

3) تطبيقات الآخرين: عادة ما تكون مفتوحة المصدر. إضافة لما ذكر في النقطتين السابقتين، فأيّ نظام لينكس يحوي برمجيات من أطراف أخرى، مما يمكن من إضافة عدد من التطبيقات المستخدمة بكثرة، سواء كان ذلك النظام الرسومي X Window نفسه، أو الخوادم كخادم الوِب أباتشي أو المتصفحات ...إلخ. وفي نفس الوقت، قد يكون من المعتاد إضافة بعض البرمجيات المملوكة، اعتماداً على مدى رغبة منشئ التوزيعة لكون البرمجيات حرة.

ولأن معظم البرمجيات حرّة أو مفتوحة المصدر، سواء كانت النواة أو برمجيات جنو أو البرمجيات الأخرى، فمن المعتاد وجود تطور سريع للإصدارات، سواء بإصلاح العلل أو إضافة مزايا جديدة. هذا يعني أنه برغبتنا إنشاء نظام جنو/لينكس، فسيكون علينا اختيار أي البرمجيات نرغب بتثبيتها في النظام، وأيّ إصدارات من تلك البرمجيات.

عالم جنو/لينكس ليس حكراً على شركة او مجتمع بعينه، مما يعني انه يتيح للجميع إنشاء أنظمتهم الخاصة المهيئة لتلبية متطلباتهم الخاصة.

دائماً ما تكون بعض تلك الإصدارات مستقرة، والبعض الآخر قيد التطوير في المرحلة الأولية (ألفا) أو الاختبارية (بيتا)، والتي قد تحوي أخطاء أو قد تكون غير مستقرة، مما يعني أنه علينا أن نكون حذرين في اختيار الإصدارات عند إنشاء نظام جنو/لينكس. وهناك مشكلة أخرى إضافية، وهي الاختيار بين البدائل، فعالم جنو/لينكس غنيّ بالبدائل، فهناك أكثر من

بديل للبرنامج الواحد. علينا الاختيار بين البدائل المختلفة، وإتاحتها كلها أو بعضها إذا كنا نرغب بأن نتيح للمستخدم حرية اختيار برامجه.

مثال:

يمكننا أخذ مدراء المكتب الذين يعملون على X Window، والذين يوفرون لنا بيئتي سطح مكتب رئيسيتين، وهما جنوم وكدي؛ لكلتيهما مزايا متشابهة وتطبيقات شبيهة أو مكملة.

في حالة موزّع أنظمة جنو/لينكس، سواء كانت تجرية أو غير ربحية، سيكون من مسؤوليته إنشاء نظام يعمل ، وذلك باختيار أفضل التطبيقات والإصدارات المتاحة.

في هذه الحالة، توزيعة جنو/لينكس هي مجموعة من البرمجيات التي تشكل نظام تشغيل مبني على نواة لينكس

ومن الحقائق المهمة التي يجب أخذها بعين الاعتبار والتي تسبب لُبساً ليس بالقليل، هي أنّه وكنتيجة لكون كل حزمة برمجيات على توزيعة معينة لها إصدارات مختلفة عن غيرها (بغض النظر عن التوزيعة)، فرقم إصدار التوزيعة لا يتماشى مع أرقام إصدارات حزم البرمجيات.

مثال:

فلنلق نظرة على بعض الإصدارات كمثال[12]:

1. نواة لينكس: يمكننا أن نجد الآن توزيعات توفر نواة واحدة أو أكثر، كالأنوية من الإصدارات القديمة من النواة 2،6, x، أو كما العادة الإصدار الحديث من النواة 3،x بمراجعة تختلف في مدى حداثتها (تعتمد على الرقم الفرعي x).
2. خيار X Window للرسوميات، بإصداراته مفتوحة المصدر، والتي يمكننا إيجادها عملياً على كلّ أنظمة جنو/لينكس، أو مشروع Xorg الحديث (تفرّع عن المشروع سابق الذكر في 2003) والأكثر شهرة، [وربما نرى مستقبلاً توزيعات تحوي إصداراً من Wayland بديلاً عن Xorg].
3. سطح المكتب أو مدير النوافذ: يمكننا إيجاد جنوم أو كدي أو كليهما؛ جنوم بإصداراتها 3،x، أو كدي بإصداراتها 4،x،y.

يمكننا على سبيل المثال إيجاد توزيعة تحوي تحوي الإصدار 3،6 من النواة، والإصدار 3،4 من جنوم؛ وتوزيعة أخرى تحوي نواة بالإصدار 3،4 وكدي بالإصدار 4،8. أيهما أفضل؟ من الصعب مقارنتها لأنها تتضمن خليطاً من المكونات، ويعتمد على كيفية عمل هذا الخليط كون المنتج النهائي أفضل أو أسوأ، أو مطوعاً أكثر أو أقل ليناسب احتياجات المستخدم. في العادة، سيبقي الموزّع توازناً بين استقرار النظام وحداثة الإصدارات المضمّنة

وبشكل عام، يمكننا تحليل أرقام الإصدارات بشكل أفضل بناء على الملاحظات التالية، والتي تحتاج لتفقّد كلٍّ منها:

1) إصدار نواة لينكس: يحدّد الإصدار بالأرقام X.Y.Z، حيث يكون X كما العادة الإصدار الرئيسيّ، والذي يعكس التغييرات الهامّة على النواة؛ أما Y فهو الإصدار الفرعيّ، وعادة ما يشير إلى تحسينات في أداء النواة: يكون الرقم زوجيًّا للإصدارات المستقرة، وفرديًّا للإصدارات قيد التطوير أو المراجعة. أما الرقم Z فهو إصدار البناء، ويشير

12 لقد تم تعديل هذه الملاحظة بشكل كبير لتحوي معلومات عن إصدارات حديثة بدل الإصدارات التي في الكتاب الأصلي من حوالي عشر سنوات

إلى رقم المراجعة للإصدار X.Y، وكل رقم يحوي ترقيعات وإصلاحات عن سابقه. لا يوفّر الموزّعون آخر إصدارات النواة، بل الإصدارات التي اختبروها أكثر والتي اختبروا استقرارها مع البرمجيات والمكونات التي يضمّنونها. نمط الترقيم التقليدي هذا (والذي بقي مستمراً منذ 2.4 وحتى الإصدارات الأولى من 2.6) قد تغيّر قليلاً ليتوافق مع كون النواة (بإصداراتها الفرعية من 2.6) تصير مستقرة أكثر، وأن مراجعاتها تقلّ (مما يعني قفزة في الرقم الأول)، ولكن التطوير مستمر بسرعة جنونية. في الخطط الأخيرة، وُجد فرع رابع من الأرقام لتحديد التغييرات الصغيرة في Z، والإمكانيات المختلفة للمراجعات (مع ترقيعات مختلفة). وبهذا تعتبر المراجعة المرقّمة بأربعة أرقام مستقرة. تُستَخدَم خطط أخرى لعدد من الإصدارات الاختبارية (وعادة لا تكون متوفرة للبيئات الإنتاجية)، وتكون متبوعة بـ rc (إصدارة مرشّحة)، أو mm وهي أنوية تجريبيّة لاختبار تقنيات مختلفة، أو git وهي نوع من الـ snapshot لتطوير النواة. تتغير أنماط الترقيم هذه باستمرار، وذلك لتتكيّف مع طريقة عمل مجتمع النواة واحتياجاته لتسريع تطوير النواة.

2) صيغة التحزيم: هذه هي الآليّة المستخدمة لتثبيت وإدارة برمجيات التوزيعة. وهي معروفة بصيغة حزم البرمجيات التي تدعمها التوزيعة. وفي هذه الحالة نجد عادة صيغ RPM و DEB و tar.gz، ورغم أن كلّ توزيعة توفر عادة إمكانية استخدام صيغ أخرى، فإن لديها صيغة مبدئيّة. عادة ما تأتي البرمجيات بملفاتها في حزمة تحوي معلومات التثبيت وحزم البرمجيات التي تعتمد عليها هذه الحزمة إن وجدت. التحزيم ضروريّ في حالة استخدام تطبيقات من طرف خارجي لا تأتي مع التوزيعة، حيث يمكن أن تكون البرمجيات متوفرة ببعض أنظمة التحزيم وليس كلها، بل وحتى يمكن أن تأتي بنظام تحزيم واحد فقط.

3) هيكلية نظام الملفات: تخبرنا هيكلية نظام الملفات الرئيسية "/" أين يمكننا إيجاد ملفاتنا (أو ملفات النظام) في نظام الملفات. لجنو/لينكس ويونكس معايير لأماكن بعض الملفات (كما سنرى في وحدة الأدوات)، مثل FHS (معيار التقسيم الهرمي لنظام الملفات). ولهذا، إذا كانت لدينا فكرة عن هذا المعيار، فسنعرف أين يمكننا إيجاد معظم الملفات؛ يلي ذلك اعتمادها على مدى التزام التوزيعة بهذه المعايير، وإذا كانت تخبرنا بأيّة تغييرات يقومون بعملها.

4) النصوص البرمجية لإقلاع النظام: تستخدم أنظمة جنو/لينكس ويونكس نصوص برمجية للإقلاع (أو shell scripts) والتي تحدد كيف يفترض أن يقلع الجهاز، والعملية (المراحل) التي ستتبع ذلك، وما يجب فعله في كل مرحلة. هناك نموذجان لهذا الإقلاع، وهما النموذج الخاص بـ SysV والنموذج الخاص بـ BSD (هذا اختلاف بين فرعي يونكس الرئيسيين)؛ ويمكن لكل توزيعة اختيار أيّ منهما. ورغم أن كلتيهما تقومان بنفس العمل، إلا أنهما يختلفان في التفاصيل، وهذا سيكون مهماً في القضايا المتعلقة بإدارة النظام (سنبحث في هذا في القسم المتعلق بالإدارة المحلية للنظام). في حالتنا، كلا النظامين الذين سنحللهما (وهما فيدورا ودبيان) يستخدمان نظام SysV (وهو ما سنراه في وحدة الإدارة المحلية)، ولكن هنالك توزيعات أخرى مثل سلاكوير تستخدم نظام BSD للإقلاع. وهناك مقترحات جديدة (مثل Upstart من أوبونتو) لخيارات جديدة لهذا النوع من الإقلاع.[13]

5) إصدارات مكتبات النظام: كلّ البرامج (أو التطبيقات) التي لدينا في النظام ستعتمد على عدد معين من مكتبات النظام لتعمل. تنقسم هذه المكتبات عادة إلى نوعين، المكتبات المرتبطة بالبرنامج بشكل ثابت (ملفات libxxx.a) والمكتبات المحملة ديناميكيًّا أثناء تشغيل البرنامج (ملفات libxxx.so). ويوفر كلا النوعين كمًّا كبيراً من الأدوات أو أكواد النظام التي تستخدمها التطبيقات. قد يعتمد عمل تطبيق على وجود مكتبات معينة، وإصدارات محددة لهذه المكتبات (غير محبّذ، لكنه قد يحدث). من الحالات الشائعة مكتبة C من جنو (المعروفة أيضاً بـ glibc). قد يطلب تطبيق ما إصداراً معيناً من glibc ليعمل أو يُصَنَّف. هذه الحالة تشكّل معضلة، ولهذا فأحد الأعمال الإصافية التي تقع على عاتق التوزيعة معرفة أيّ إصدارات glibc توفر، والإصدارات الأخرى المحتملة والمتوافقة مع الإصدارات الأقدم. تظهر المشكلة عند محاولة تشغيل أو تصنيف منتج برمجي قديم على توزيعة حديثة، أو منتج برمجي حديث جدًّا على توزيعة قديمة.

حدثت أكبر التغيرات عند الانتقال إلى glibc 2.0، والتي فرضت إعادة تصنيف كلّ البرامج لتعمل بشكل صحيح،

---

13 معظم التوزيعات الحديثة تتبع نموذج systemd المبني على نموذج SysV والمتوافق معه، والذي طوره فريق فيدورا ليدعم تشغيل الخدمات على التوازي ليقلع النظام بشكل أسرع. من الأمثل الأخرى على الإقلاع المتوازي نظام Upstart الذي طورته أوبونتو قبله، والذي تستخدمه دبيان للآن، وذلك لإمكانية استخدامه مع نواة bsd، عكس نظام systemd والذي لا يعمل إلا على لينكس.

وفي المراجعات المختلفة المرقمة 2,x، كانت هناك تغييرات طفيفة يمكنها التأثير على عمل برنامج ما. في حالات كثيرة، تتفحّص حزم البرمجيات توفر الإصدار المناسب من glibc، أو يحوي اسمها رقم الإصدار الذي تحتاجه (مثال: package-xxx-glibc2.rpm).

6) سطح مكتب X Window: نظام X Window معيار لرسوميات أسطح المكتب في جنو/لينكس. طورته MIT عام 1984، وكل أنظمة يونكس تحوي فعليًّا إحدى إصداراته. لتوزيعات جنو/لينكس إصدارات مختلفة مثل Xfree86 [الذي عفا عليه الزمن] و Xorg. عادة ما يكون X Window طبقة رسوميات وسيطة تعتمد على طبقة أخرى تعرف بمدير النوافذ لإظهار عناصرها. يمكننا أيضاً ضمّ مدير النوافذ إلى عدد من التطبيقات والأدوات للحصول على ما يسمى ببيئة سطح المكتب.

للينكس بيئتا سطح مكتب أساسيّتان، وهما جنوم وكدي. ما يميز كلًّا منهما هو كونها مبنية على مكتبة من مكوناتها الخاصة (العناصر المختلفة للبيئة، كالنوافذ والأزرار والقوائم وغيرها): جتك+ (gtk+) في جنوم، وكيوتي (QT) في كدي مكتبتا الرسوميات الرئيسيتان المستخدمتان لبرمجة التطبيقات في هاتين البيئتين. إضافة لهاتين البيئتين، هناك العديد من مدراء النوافذ وأسطح المكتب، مثل: XFCE, Motif, Enlightenment, BlackIce, FVWM وغيرها، مما يعني وجود طيف واسع من الخيارات. إضافة لهذا، تتيح كل منها إمكانية تغيير مظهر النوافذ والمكونات حسب رغبة المستخدم، او حتى إمكانية عمل مظاهر جديدة حسب الرغبة.

7) برمجيات المستخدم: البرمجيات التي يضيفها الموزِّع، وعادة ما تكون مفتوحة المصدر، للمهام الشائعة (أو غير الشائعة جدًّا، أو التي تصبّ في مجالات متخصصة جدًّا). التوزيعات الشائعة ضخمة جداً، حيث يمكننا إيجاد مئات وحتى آلاف من التطبيقات الإضافيّة (للعديد من التوزيعات من قرص ضوئي واحد وحتى 50 قرصاً فأكثر، أي ما يعادل حوالي ثمانية أقراص DVD، أو قرصين BlueRay).

تغطي هذه التطبيقات عملياً كلّ المجالات، سواء الاستخدام المنزلي أو الإداري أو العلمي. وتضيف بعض التوزيعات برمجيات مملوكة لأطراف خارجية، أو برمجيات خادم أعدّها الموزِّع، مثل خادم بريد، أو خادم وِب

آمِن، أو ما إلى ذلك.

هكذا يطلق الموزعون إصدارات مختلفة من توزيعاتهم، فمثلاً تكون هناك في بعض الأحيان اختلافات بين الإصدار الشخصيّ والاحترافيّ وإصدار الخوادم.

عادة ما تكون هذه التكلفة المادية غير منطقية، لأن البرمجيات القياسية كافية (مع قليل من العمل الإداري الإضافيّ)؛ لكنها قد تكون جذابة للشركات، لأنها تقلل الوقت اللازم لتثبيت الخادم، وكذلك تقلل من الصيانة والمتابعة، كما وتحسّن بعض الخوادم والتطبيقات الهامّة لإدارة تقنية المعلومات في الشركة.

6.1 دبيان



شكل 2: شعار دبيان

دبيان حالة خاصة، حيث أنها توزيعة قائمة على مجتمع ليس له أهداف تجارية غير الاستمرار في توزيعها والتسويق لاستخدام البرمجيات الحرة ومفتوحة المصدر.

دبيان توزيعة يدعمها مجتمع متحمّس من مستخدميها ومطوريها قائم على الالتزام باستخدام برمجيات حرة.

تأسس مشروع دبيان عام 1993، وذلك لإنشاء توزيعة دبيان جنو/لينكس. صارت بعد ذلك مشهورة، بل وحتى تنافس توزيعات تجارية من حيث الاستخدام، مثل ردهات وماندريك. ولأنها مشروع مجتمعيّ، فتطوير هذه التوزيعة مضبوط بسلسلة من السياسات والقوانين؛ هناك وثائق تعرف "بعقد دبيان الاجتماعي" تذكر الفلسفة العامة للمشروع وسياسات دبيان محدّدة بالتفاصيل كيفيّة تنفيذ توزيعتها.

أهداف دبيان مرتبطة ارتباطاً وثيقاً بأهداف مؤسسة البرمجيات الحرّة ومشروع البرمجيات الحرة جنو التابع لها؛ ولهذا السبب، دائماً يضمنون "دبيان جنو/لينكس" في اسم توزيعتهم؛ إضافة إلى هذا، فقد ساهم نصّ عقدهم الاجتماعيّ بوضع الأساس الذي قام عليه تعريف البرمجيات مفتوحة المصدر. وما يهمّ في سياساتهم هو أنّ على أيّ شخص يرغب بالمشاركة في مشروع التوزيعة الالتزام بها. رغم أني لست من فريق التوزيعة، إلا أن هذه السياسات مثيرة للاهتمام لأنها تشرح كيف تعمل توزيعة دبيان.

علينا أن نذكر أيضاً جانباً عملياً يهم المستخدمين: لقد كانت دبيان دائماً توزيعة صعبة. دبيان هي توزيعة لينكس التي يستخدمها الهاكرز، أي هؤلاء الذي يحصلون على النواة ويعدلونها، ومبرمجي اللغات الدنيا، والذين يرغبون بأن يكونوا في المقدمة لاختبار البرمجيات الحديثة، ولاختبار تطويرات النواة غير المنشورة ... وبكلمات أخرى، كلّ أنواع الناس المهتمين بجنو/لينكس إلى درجة الجنون.[14]

---

14 ما يقصده الكاتب بالجنون هنا هو الهوس. قد يكون كلامه هنا مبالغاً فيه حسبما أسمع من بعض مستخدمي دبيان، عدا عن كون الكتّاب قديماً نسبيًّا = وأن الأمور قد تكون تغيرت الآن، إلا أن فيه جانباً من الحقيقة، خاصة في ما يتعلق بالتثبيت النصّيّ والذي لا يريح المبتدئين، وكون =

اشتهرت الإصدارات الأولى من دبيان بصعوبة تثبيتها. الحقيقة هي أنه لم يكن قد تمّ العمل كفاية على جعلها سهلة التثبيت لغير الخبراء. لكن الأمور تحسنت مع الوقت. فالآن يحتاج التثبيت لبعض المعرفة، لكن يمكن أن يتم باتباع القوائم (القوائم النصية، وليس قوائم رسومية كالتي تاتي مع بعض التوزيعات الأخرى). وهناك برامج لتسهيل تثبيت الحزم. ومع هذا، فقد يُصدم المستخدم عند أول محاولة تثبيت.

عادة ما تتوفر تنويعات (تدعى نكهات) لتوزيعة دبيان. حاليًّا، هناك ثلاثة فروع للتوزيعة: المستقر، والاختباري، وغير المستقر. وكما تشير أسماؤها، المستقر هو المستخدم للبيئات الإنتاجية (أو المستخدمين الذين يرغبون بالاستقرار)، والاختباري يوفر برمجيات جديدة تم اختبارها قليلاً (يمكننا اعتبارها إصدار بيتا لدبيان) سيتم تضمينها في الفرق المستقر قريباً، أما الفرع غير المستقر، فيوفّر أحدث البرمجيات وحزمها تتغير في وقت قصير؛ حيث يمكن أن تتغير عدّة حزم خلال أسبوع، بل وحتى خلال يوم واحد. كلّ التوزيعات يمكن تحديثها من أكثر من مصدر (CD, FTP, web)، أو عبر نظام يسمّى APT يدير حزم برمجيات دبيان المسماة DEB. للتوزيعات الثلاثة أسماء شائعة أكثر، فمثلاً [(في وقت ما من تاريخ دبيان) كانت كالتالي]:

1) المستقرة Etch
2) التجريبية Lenny
3) غير المستقرة Sid

الإصدار المستقر السابق له كان اسمه Sarge – 3.1r6، والذي سبقه Woody – 3.0. والأحدث عام 2007 هو دبيان جو/لينكس Etch – 4.0. الإصدارات الأطول دعماً هما الأكثر والأقل استقراراً. في هذا الوقت [أي وقت تأليف هذا الكتاب] Sid غير محبّذ لبيئات العمل اليوميّ (الإنتاجي)، وذلك لأنها قد تحوي مزايا لم ينته اختبارها بعد، وقد تفشل (رغم أن هذا غير شائع)؛ هي توزيعة جنو/لينكس التي يستخدمها الهاكرز. هذا الإصدار يتغير تقريباً بشكل يوميّ؛ هذا عاديّ إذا كان هناك تحديثات يريدون إضافتها، لأن هناك 10 إلى 20 حزمة برمجيات جديدة يوميًّا (أو حتى أكثر في بعض أوقات التطوير).

من المرجح أن Etch أفضل الخيارات لبيئات العمل اليومي، فهي تحدّث دوريًّا لتوفر البرمجيات الجديدة أو التحديثات (كالتحديثات الأمنية). عادة لا تحوي أحدث البرمجيات، والتي لا يتم تضمينها حتى يخضعها المجتمع لعدد كبير من

المستودعات الرئيسية وأقراص التثبيت لا تحوي مشغلات بعض بطاقات العرض المنتشرة.

الاختبارات.

سنعلّق بإيجاز على بعض مزايا هذه التوزيعة (الإصدارات الافتراضية الحالية لكل من المستقرة وغير المستقرة:

1) الإصدار الحالي (المستقر) يتكون من 1 – 53 قرص ضوئي مضغوط (أو 8 أقراص DVD) من أحدث إصدار متاح من دبيان. عادة ما يكون هناك عدد من الإمكانيات، ويعتمد ذلك على البرمجيات التي يمكننا إيجادها على الوسائط الملموسة (الأقراص الضوئيّة المضغوطة والأقراص الرقمية متعددة الاستخدامات)، أو التي يمكننا تنزيلها لاحقاً من الإنترنت، والتي نحتاج من أجلها قرصاً ضوئيًّا بسيطاً (وسيط التثبيت عبر الشبكة – netinstall CD)، إضافة إلى الوصول إلى الإنترنت لتنزيل البقيّة عند الحاجة. يمكن شراء هذه التوزيعة (بمقابل رمزي يكفي لتغطية تكاليف الموارد، مما يساهم في استمرار التوزيعة) أو يمكن تنزيلها من debian.org أو مراياها.

2) الإصدارات الاختبارية وغير المستقرة ليس لها أقراص رسميّة، لكن يمكن تحويل النسخة المستقرّة إلى اختبارية أو غير مستقرة بتغيير إعدادات نظام الحزم ATP.

3) نواة لينكس: كانت الأنوية الافتراضية 2٫6٫x، وكانت 3٫x اختيارية، أما الآن في الإصدارات الأخيرة، فقد صارت 3٫x الخيار المبدئيّ. تركيز دبيان المستقرة على دعم الاستقرار وإتاحة الخيار للمستخدمين لمنتجات برمجية أخرى أحدث إذا أرادوا ذلك (في الاختبارية أو غير المستقرة).

4) صيغة التحزيم: تدعم دبيان إحدى الصيغ التي توفر معظم التسهيلات، وهي APT. لحزم البرمجيات صيغة تعرف بـ DEB. أما APT فهي أداة عالية المستوى لإدارة هذه الحزم ومتابعة قاعدة بيانات للحزم التي يمكن تثبيتها فوراً أو الحزم المتوفرة. يمكن لـ APT أيضاً الحصول على برمجيات من مصادر عديدة: CD, FTP, web.

5) يمكن تحديث نظام APT في أيّ وقت، وذلك عبر قائمة من مصادر برمجيات دبيان (APT sources)، والتي قد تحوي المستودع المبدئي لدبيان (debian.org) أو مواقع لأطراف خارجية. بهذه الطريقة لا ترتبط التوزيعة بشركة وحيدة أو بنظام اشتراك مدفوع وحيد.

6) من الحزم المستخدمة مثلاً eglibc و Xorg.

7) لسطح المكتب، تقبل جنوم 3٫6 (المبدئي) أو كدي 4٫8. أما غير المستقرة فتقبل 3٫7٫9 أو كدي 4٫10.

8) أما من حيث التطبيقات المثيرة للاهتمام، فتحوي معظم التطبيقات التي يمكننا إيجادها في توزيعات جنو/لينكس، في Sid: توجد محررات مثل emacs (و xemacs)، ومُصرِّف وأدوات gcc، وخادم الوِب أباتشي، ومتصفح موزيلا فيرفكس، وبرنامج سامبا لمشاركة الملفات مع وندوز، وغيرها.

9) تحوي أيضاً حزماً مكتبية مثل المكتب الحرّ و KOffice.

10) تحوي دبيان العديد من ملفات الإعداد المخصصة للتوزيعة في المجلدات الموجودة تحت /etc/.

11) تستخدم دبيان مدير الإقلاع grub-legacy بشكل مبدئيّ، لكن يمكنها أيضاً استخدام grub2.

12) الإعداد للاستماع إلى خدمات الشبكة عبر ميفاق TCP/IP، والذي يتم عبر خادم inetd كما في معظم أنظمة يونكس، (والذي يمكن ضبطه من ملف الإعداد /etc/inetd.conf). إضافة إلى هذا، فهي تحوي خياراً إضافيًّا، وهو xinetd، والذي بدأ يتحول إلى الخيار المفضّل.

13) هناك العديد من توزيعات جنو/لينكس المبنيّة على دبيان، حيث يمكن تطويع النظام بسهولة لعمل توزيعات أكبر أو أصغر مع برمجيات أكثر أو أقل تمّ تطويعها لغرض معيّن. من التوزيعات المعروفة منها نوبِكْس Knoppix، وهي توزيعة من قرص ضوئيّ واحد حيّ (تعمل من القرص الضوئي مباشرة دون تثبيت)، والشائع استخدامها في استعراض جنو/لينكس، أو لاختباره على جهاز دون تثبيت، وذلك لأنها تعمل من القرص الضوئيّ، عدا عن إمكانية تثبيتها على القرص لتصير توزيعة دبيان قياسية. Linex توزيعة أخرى صارت مشهورة نوعاً ما وذلك بسبب تطويرها المدعوم من السلطة المحلية للمجتمع المستقل لإكستريمادورا. وفي نفس الوقت نجد أوبونتو وهي إحدى التوزيعات التي حققت أكبر إنجاز للتوزيعات الدبيانية (وحتى أنها تخطت دبيان في بعض الأمور)، وذلك لسهولتها في بناء نظام سطح مكتب بديل.

ملاحظة:

يمكن استخدام دبيان كأساس لتوزيعات أخرى؛ فتوزيعة نويكس على سبيل المثال مبنية على دبيان ليتمكن المستخدم من تشغيلها من القرص الضوئي دون الحاجة لتثبيتها على القرص الصلب. و linex توزيعة دبيانية تم تطويعها لتناسب المجتمع المستقل لإكستريمادورا كجزء من مشروعها لتبني البرمجيات مفتوحة المصدر. وأوبونتو توزيعة محسنة لبيئات سطح المكتب.



شكل 3: بيئة دبيان Sid مع جنوم 2.14

6.2 فيدورا كور

شركة ردهات Red Hat Inc من الشركات التجارية الرئيسية في عالم جنو/لينكس، ولديها إحدى أكثر التوزيعات نجاحاً. أنشأ بوب يَنج Bob Young و مارك إيوينج Marc Ewing شركة ردهات عام 1994. لقد كانا مهتمين بنماذج البرمجيات مفتوحة المصدر وفكرا بأنها يمكن أن تكون طريقة جيدة للعمل التجاري. منتجهما الرئيسي توزيعة ردهات لينكس (والتي سنشير اليها اختصاراً: ردهات)، والمتاحة لقطاعات مختلفة من السوق كالمستخدمين المنفردين (إصدارات شخصية واحترافية) أو الشركات المتوسطة أو الكبيرة (بالإصدارة التجارية Enterprise، وإصداراتها الفرعية المختلفة).

> ردهات لينكس هي توزيعة لينكس التجارية الرئيسية، وموجهة لكل من الأجهزة المكتبية الشخصية، وقطاع عريض من أسواق الخوادم. إضافة إلى ذلك، فشركة ردهات من أكثر الشركات مساهمة في تطوير لينكس، حيث العديد من الأعضاء المهمين من مجتمع لينكس يعملون فيها.





شكل 4: شعار فيدورا على اليمين وردهات على اليسار

رغم أن ردهات تعمل بنموذج برمجيات مفتوحة المصدر، إلا أنها شركة ذات أهداف ربحية، وهذا سبب إضافتهم قيمة إلى توزيعتهم البسيطة عبر عقود الدعم، واشتراكات التحديثات، ووسائل أخرى. بالنسبة للشركات، فهم يضيفون برمجيات مفصّلة (أو برمجيات خاصة بهم)، وذلك لتطويعها لمجال عمل الشركة، سواء باستخدام خوادم محسنة، أو برمجيات أدوات تملكها ردهات.

في مرحلة ما (وتحديداً نهاية عام 2003) قررت ردهات لينكس (الإصدار 9.x) إيقاف إصدارها لسطح المكتب

لجنو/لينكس، ونصحت عملاءها بالانتقال إلى الإصدار الموجّه لقطاع الأعمال، والتي ستبقى الإصدار الوحيد المدعوم رسميًّا.

وفي ذلك الوقت، قررت ردهات بدء المشروع المفتوح للمجتمع والمعروف بفيدورا، والتي تحمل وجهة إنتاج توزيعة يقوم عليها المجتمع (على نمط دبيان، لكن لأهداف أخرى)، ولتسمى فيدورا كور. في الحقيقة، الهدف من ورائها إنشاء مختبر تطوير مفتوح للمجتمع مما يجعل بالإمكان اختبار التوزيعة وفي نفس الوقت توجيه التطوير التجاري للشركة في توزيعتها الموجهة للأعمال.

إلى حد ما، أشار المنتقدون إلى أنه يتم استغلال المجتمع كمختبرين للإصدارات الأولية للتقنيات التي ستضمّن في المنتجات التجارية لاحقاً. هذا النموذج أيضاً مستخدم في شركات أخرى لإنشاء نموذج مزدوج لتوزيعات مجتمعية وتجارية. كمثال على ذلك توزيعة أوبن سوزة (المبنية على توزيعة سوزة التجارية)، و freespire (المبنية على Linspire).

في العادة، يقدم النموذج المزدوج لردهات ومجتمع فيدورا رؤية محافظة معينة (أقل وضوحاً في فيدورا) في عناصر البرمجيات التي تضيفها إلى التوزيعة، حيث سوقها الرئيسي هو الأعمال، وتحاول جعل توزيعتها مستقرة قدر الإمكان، حتى وإن عنى ذلك عدم إتاحة آخر التحديثات. إنّ ما تقدمه كقيمة مضافة تنقيحها المستمر لنواة لينكس في توزيعتها، وإنشاء تصحيحات وترقيعات لتحسين الاستقرار. حتى أنها قد تعطّل ميزة أو مشغل عتاد ما في النواة، إذا اعتبرت أنه غير مستقر كفاية. توفر أيضاً العديد من الأدوات في البيئات الرسومية والبرامج الرسومية الخاصة بها، وتستخدم كلتي بيئتي سطح المكتب جنوم (المبدئي) وكدي، ولكن ضمن بيئتها المعدلة المسمّاة BlueCurve، والتي تجعل كلي سطحي المكتب متماثلين فعلياً (النوافذ والقوائم، وغيرها).

الإصدار الذي سنستخدمه سيكون أحدث إصدار متاح من فيدورا كور، والتي سنطلق عليها كلمة فيدورا للتسهيل. بشكل عام، التطويرات والمزايا التي تصان تبقى مشابهة تقريباً في الإصدارات التي تطلق بعدها، مما يعني أن معظم التعليقات ستكون صالحة للإصدارات المختلفة عبر الزمن. علينا الأخذ بعين الاعتبار أن مجتمع فيدورا يحاول الالتزام بجدول زمني مدته ستة شهور تقريباً لكل إصدار. وهناك إجماع على المزايا الجديدة التي تضاف في كلّ إصدار.

ردهات – من ناحية أخرى – تترك إصدارات سطح المكتب بأيدي المجتمع وتركز نشاطها على إصدار الأعمال (

.(Red Hat Enterprise WS, ES & AS

فلنلقِ نظرة على بعض مزايا توزيعة فيدورا كور بإيجاز:

1) يتكون الإصدار الحالي من عدّة أقراص ضوئية مضغوطة، أولها قرص إقلاع يفيد في التثبيت. هناك أيضاً أقراص إضافيّة تحوي التوثيق والمصدر البرمجي لمعظم البرمجيات المثبتة في التوزيعة. تتوفر التوزيعة أيضاً على أقراص DVD.

2) نواة لينكس: تستخدم أنوية من سلسلة 3,x يمكن تحديثها عبر نظام حزم rpm، وذلك باستخدام yum مثلاً (راجع الجزء المتعلق بالنواة). تُخضِع ردهات النواةَ إلى اختبارات كثيرة وتنتج ترقيعات لإصلاح المشاكل، والتي في العادة تدمج في نسخة مجتمع النواة، حيث العديد من المساهمين الهامين في لينكس يعملون في ردهات أيضاً.

3) هيئة التحزيم: تنشر ردهات برمجياتها بنظام تحزيم RPM – Redhat Package Manager، والتي تُدار بالأمر rpm أو بأدوات yum (وسنعلّق على هذا في الجزء المتعلق بالإدارة المحلية للنظام). RPM أحد أفضل أنظمة التحزيم المتاحة (وينافسه DEB من دبيان)، وبعض أنظمة يونكس المملوكة تحويه أيضاً. ببساطة، لنظام RPM قاعدة بيانات صغيرة بالحزم المثبتة، وتحقق من كون الحزمة المراد تثبيتها عبر rpm ليست مثبتة مسبقاً، وأنها لا تتعارض مع أيّ حزمة برمجيات أخرى مثبتة، وأن حزم البرمجيات المطلوبة (الاعتمادية) موجودة، وأن الإصدار المناسب مثبت. حزمة RPM بالأساس مجموعة من الملفات المضغوطة وتحوي معلومات عن الاعتماديات أو البرمجيات التي تحتاجها.

4) بالنسبة للإقلاع، تستخدم فيدورا النصوص البرمجية من نوع System V (والتي سننظر فيها في الجزء المتعلق بالإدارة المحلية).[15]

5) من الحزم المستخدمة: Xorg و glibc وغيرها.

15 تستخدم فيدورا حالياً systemd وليس النصوص البرمجية المتعلقة بنظام System V نفسها. يوفر systemd سرعة أكبر بالإقلاع وذلك بتحميل الخدمات بشكل متوازي بدل التسلسل، ويوفر متابعة أفضل للخدمات التي تعمل، كما ويوفر أيضاً توافقية كاملة مع نصوص System V للإقلاع.

6) سطح المكتب يمكن أن يكون جنوم (المبدئي) أو كدي اختيارياً.

7) عند الحديث عن التطبيقات الهامة، تحوي فيدورا معظم التطبيقات التي يمكننا أن نجدها في كل توزيعات جنو/لينكس تقريباً: المحررات (مثل emacs)، مُصرِّف وأدوات gcc، خادم الوِب أباتشي، متصفح الوِب موزيلا/فيرفكس، برمجيات سامبا لمشاركة الملفات مع وندوز، وغيرها.

8) تحوي فيدورا أيضاً برمجيات مكتب مثل المكتب المفتوح و KOffice.

9) يمكن الحصول على برمجيات إضافية عبر خدمات تحديث yum (كما في غيرها) بطريقة مشابهة لنظام APT في دبيان، أو باستخدام أدوات تحديث مختلفة، أو من الإنترنت وذلك بتنزيل حزم RPM مصممة للتوزيعة.

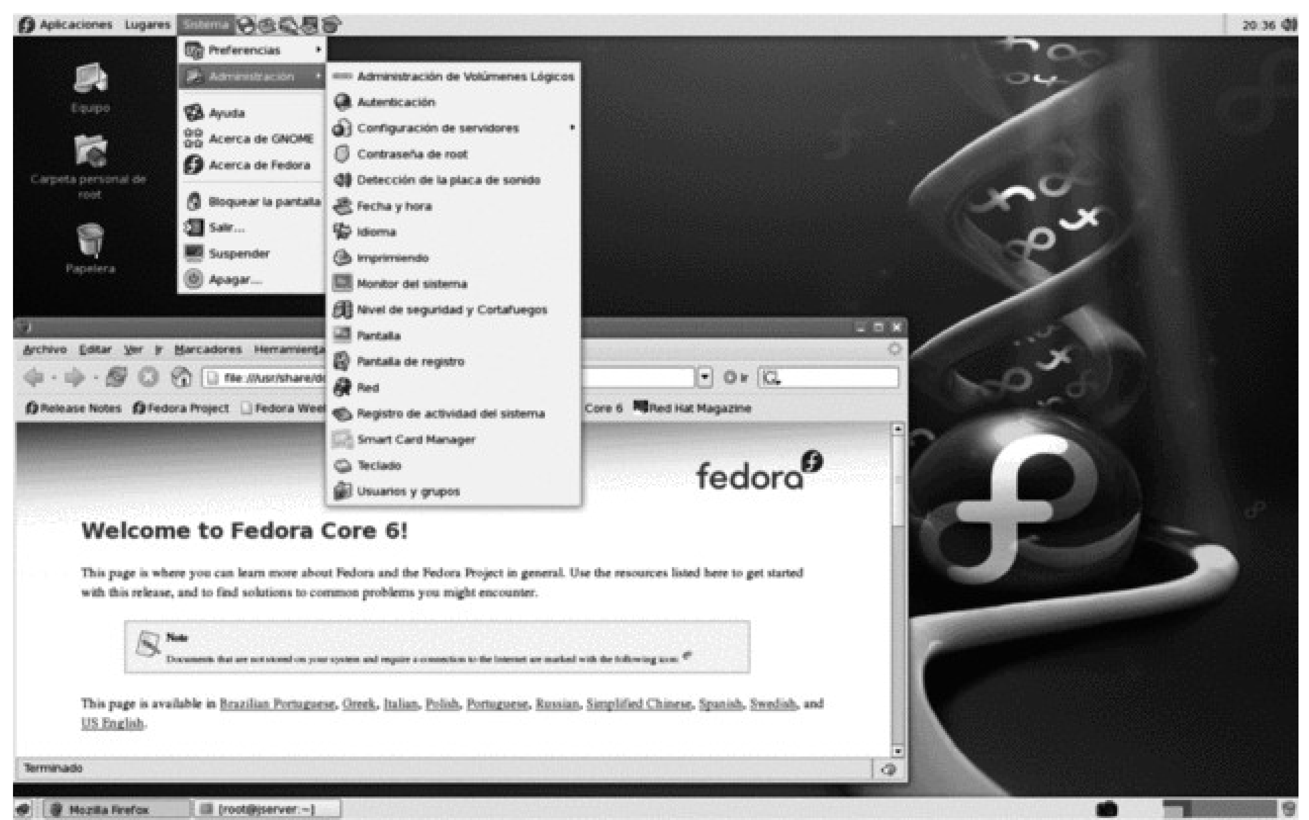
10) تستخدم فيدورا محمل إقلاع Grub مبدئيًّا.

11) استبدلت ردهات إعداد الاستماع إلى خدمات شبكة TCP/IP xinetd، مكان inetd (إعداده في الملف /etc/inetd/conf) الذي تستخدمه معظم أنظمة يونكس. يميز xinetd الإعداد المقسّم إلى وحدات (الإعدادات في المجلد /etc/xinetd.d).

12) كان فيها قديماً برنامج اسمه kudzu يعمل عند الإقلاع ليتفحص تغييرات العتاد ويكتشف العتاد المثبت حديثاً. وقد تم الاستغناء عنه بسبب وجود واجهة برمجة تطبيقات API جديدة اسمها HAL تقوم بهذه المهمّة، التي ما لبثت أن حلّ مكانها نظام udev المستخدم حالياً.

13) هناك العديد من التوزيعات الأخرى المبنية على ردهات أيضاً، والتي تحتفظ بالعديد من مزاياها، وتحديداً ماندريفا (والتي كانت تسمى Mandrake): وهي توزيعة فرنسية كانت بالأصل مبنية على ردهات، وقد بقيت الاثنتان في مقدمة خيارات المستخدمين (خاصة فيما يتعلق بسطح المكتب). تطور ماندريفا برمجياتها الخاصة والعديد من المعالجات للمساعدة في إدارة وتثبيت معظم المهام الشائعة، مما يبعدها عن أصلها الذي بدأ بالبناء على ردهات. وفي نفس الوقت، ساهمت إصدارات الأعمال من توزيعة ردهات في بروز سلسلة من التوزيعات المجانية ذات الشعبية الكبيرة المهتمة ببيئة الخوادم، مثل سنتوس

(والتي تحاول الإبقاء على توافقية كاملة مع إصدار ردهات للأعمال)، و Scientific Linux (متخصصة في الحسابات العلمية لمشاريع البحث العلمي). أما فيما يتعلق بأنظمة التحزيم، فما يسترعي الاهتمام هو أن نظام RPM مستخدم في عدد كبير من التوزيعات، ومن ضمنها SuSe.



شكل 5: سطح مكتب فيدورا بواجهة جنوم

فيما يتعلق بتوزيعة المجتمع فيدورا كور، وأصولها التجارية في ردهات:

1) هي توزيعة يُنشئها مجتمع من المبرمجين والمستخدمين ويقومون على تطويرها؛ لا يوفر المنتج لها أي دعم للتحديثات أو الصيانة. هذا يجعلها معتمدة على المجتمع، كما هي حال توزيعة دبيان جنو/لينكس.

2) تُنتَج هذه الإصدارات بشكل سريع نسبيًّا، والإصدارات الجديدة من التوزيعة تأتي كلّ ستة أشهر تقريباً.

3) تستخدم نظام تحزيم RPM أيضًا. فيما يتعلق بعملية تحديث حزم التوزيعة وتثبيت حزم أخرى جديدة، فيمكن

عملها باستخدام أدوات مختلفة، وغالباً ما يتم استخدام yum فقط في الإصدارات الحديثة منها.[16]

4) يمكن إيجاد أمور أخرى أكثر تقنية (سنبحث فيها في الأجزاء اللاحقة من هذا الكتاب) في ملاحظات الإصدار.

---

16 تم تعديل هذه النقطة لتحديث معلوماتها، وتجنب تشتيت القارئ بأمور تاريخية.

7 ما سيغطيه هذا الكتاب ...

بعد دراستنا لهذه المقدمة "الفلسفية" لعالم المصدر المفتوح وتاريخ أنظمة يونكس وجنو/لينكس، إضافة إلى توضيح مهام مدير النظام، سنبحث في طرق التعامل مع المهام الاعتيادية المتعلقة بإدارة أنظمة جنو/لينكس.

فيما يلي، سنلقي نظرة على النواحي المختلفة المتعلقة بإدارة أنظمة جنو/لينكس. وفي كلّ ناحية، سنحاول الخوض في بعض الأساسيات النظرية البسيطة التي ستساعدنا على شرح المهام التي يجب عملها وفهم كيف تعمل الأدوات التي سنستخدمها في العمل. سيُتبَع كل موضوع بنوع من الشرح العملي الذي سيغطي جلسة عمل صغيرة أو كيفية استخدام بعض الادوات. علينا أن نتذكر أيضاً ما سبق وقلناه في المقدمة، وهو أن مجال إدارة الأنظمة واسع جدًّا، وأيّ محاولة لتغطيته بشكل كامل ستبوء بالفشل، وذلك بسبب الحجم المحدود للكتّاب؛ ولهذا، فهناك الكثير من المراجع (من كتب، وصفحات، ومواقع، وشروحات، وغيرها) حول كلّ موضوع، والتي ستساعدك على تعميق المعرفة التي لديك والتي ستكون قد اكتسبتها من المقدمة الموجزة حول الموضوع.

المواضيع التي سنبحث فيها كالآتي:

1) في الجزء المتعلق بالهجرة بين الأنظمة، ستتكون لدينا وجهة نظر حول أنظمة الحاسوب التي نستخدمها وفي أيّ بيئة عمل تُستَخدَم؛ سنرى أيضاً كيف تتأقلم أنظمة جنو/لينكس أفضل أو أقل في كلّ واحدة منها، وسنتعامل مع أول معضلة تتعلق بتقديم نظام جنو/لينكس: هل نغير النظام الذي لدينا؟ أم ننتقل تدريجياً بوجود النظامين القديم والجديد؟

2) في الجزء الثاني، سندرس الأدوات التي على مدير النظام أن يعيش حياته اليومية معها، والتي يمكن أن تشكل ما يشبه صندوق الأدوات لمدير النظام. سنتحدث عن معايير جنو/لينكس، والتي ستسمح لنا بتعلم النواحي المشتركة في كلّ توزيعات جنو/لينكس. وبعبارة أخرى، ما يمكننا توقعه في أيّ واحدة منها.

3) في الجزء المتعلق بالنواة، سنتعمّق أكثر في نواة لينكس، وسنرى كيف يمكننا ضبطها بشكل أفضل للعتاد أو الخدمات التي ستتوفر في نظامنا، وذلك بتفصيلها على مقاسنا.

4) في جزء الإدارة المحلية، سنتعامل مع النواحي الإدارية التي يمكننا اعتبارها "محلّيّة" في نظامنا. يمكن أن تشكل هذه النواحي معظم النواحي الإدارية الاعتيادية، وهي تتعلق بالتعامل مع عناصر مثل المستخدمين والطابعات والأقراص والبرمجيات والعمليات وغيرها.

5) في الجزء المتعلق بالشبكة، سنبحث في كل المواضيع المتعلقة بنظامنا والأنظمة المجاورة على الشبكة بغض النظر عن نوعها، وسننظر في الأنواع المختلفة للاتصال والتي يمكن أن توجد بين نظامنا والأنظمة المجاورة له على الشبكة أو الخدمات التي يمكننا الاستفادة منها أو تقديمها لهذه الأنظمة.

6) في الجزء المتعلق بالخوادم، سننظر في بعض الإعدادات الاعتيادية للخوادم والتي يمكننا إيجادها على نظام جنو/لينكس عادة.

7) في الجزء المتعلق بالبيانات، سنبحث في إحدى أهم القضايا ذات العلاقة بالموضوع هذه الأيام، وهي آليات تخزين وتبادل البيانات التي يمكن لأنظمة جنو/لينكس تقديمها لنا، وبالتحديد أنظمة قواعد البيانات وآليات إدارة الإصدارات.

8) في الجزء المتعلق بالأمن، سنتعامل مع إحدى أهم وأكثر القضايا المتعلقة بأمن نظام جنو/لينكس هذه الأيام. إن وجود عالَم متشابك عبر الإنترنت لَيؤدي إلى وجود سلسلة من المخاطر الكبيرة على العمل الصحيح لنظامنا، وتُنبئ بوجود مشاكل تتعلق بالموثوقية لكلّ من الأنظمة والبيانات التي قد نتلقاها أو نتيحها عبر الشبكة. ولهذا، فعلى أنظمتنا توفير حدّ أدنى من الأمن ومنع الوصول غير المصرّح به إلى البيانات أو التعامل معها. سنبحث في أكثر أنواع الهجمات شيوعاً، والسياسات الأمنية التي يمكن تطبيقها، والأدوات التي يمكنها مساعدتنا على التحكم بمدى أمن نظامنا.

9) في الجزء المتعلق بالتحسينات، سنرى بأن لأنظمة جنو/لينكس عدد كبير من المتغيرات المفعلة والتي يمكن أن تؤثر على أداء التطبيقات أو الخدمات التي نوفرها، وذلك بسبب العدد الكبير للخوادم والخدمات التي توفرها، إضافة إلى العدد الكبير من البيئات التي صمّم النظام لتغطيتها. يمكننا، أو بالأحرى علينا أن نستخرج أفضل أداء

وذلك بتحليل إعدادات النظام نفسه وضبطها لتتناسب مع جودة الخدمة التي نرغب بتوفيرها لزبائننا.

10) وفي الجزء الأخير المتعلق بالشبكات العنقودية، سنبحث في بعض التقنيات المستخدمة لتوفير حوسبة عالية الأداء على أنظمة جنو/لينكس، والمستخدمة كثيراً في مجالات الحوسبة العلمية، والتي أصبحت مستخدمة بكثرة في عدد كبير من المجالات الصناعية (كالصيدلة والصناعات الدوائية، والكيمياء، والصناعات العسكرية، وغيرها)، وذلك للأبحاث وتطوير منتجات جديدة. إضافة إلى تنظيم العديد من أنظمة جنو/لينكس في شبكات عنقودية لمضاعفة أداء الأنظمة المنفردة، وذلك بإنشاء مجموعات من الأنظمة تجعل من الممكن الصعود بمستوى الخدمات المقدمة لتلبي الحاجة المتزايدة للزبائن.

أنشطة:

1) اقرأ البيان الرسمي لدبيان على:

http://www.debian.org/social_contract

2) اقرأ عن التوزيعات المختلفة المبنية على دبيان: نكهات نوبِكس و linex و أبونتو. إضافة إلى الموقع الرسمي لهذه التوزيعات، يوفر موقع www.distrowatch.com دليلاً جيداً للتوزيعات وحالتها إضافة إلى البرمجيات التي تتضمّنها. يمكننا الحصول على صور ISO لمختلف التوزيعات عبر هذا الموقع أو بالوصول إلى المواقع المختلفة للمجتمعات والمُصَنّعين.

المراجع:

لمصادر ومراجع أخرى حول الموضوع، انظر إلى قائمة المراجع في نهاية الكتاب.

[LDP] مشروع توثيق لينكس The Linux Documentation Project – LDP، مجموعة من الأدلة والشروحات التي تغطي كل نواحي جنو/لينكس.

[ODSb] مجتمع له العديد من المواقع والأخبار والتطويرات والمشاريع وغيرها.

[Sla] موقع اخبار مجتمع المصدر المفتوح ومواقع عامة عن الإنترنت وتكنولوجيا المعلومات.

[Bar] [New] أخبار المصدر المفتوح Open source News.

[Sou] [Fre] قائمة مشاريع مفتوحة المصدر.

[Dis] مراقب توزيعات جنو/لينكس والمزايا الجديدة لحزم البرمجيات. وروابط لمواقع تنزيل صور ISO للأقراص المضغوطة ومتعددة الطبقات لتوزيعات جنو/لينكس.

[His][Bul][LDP] توثيق عام ومجتمعات مستخدمين.

[Mag03][Jou03] مجلات جنو/لينكس.

الهجرة
والتواجد المشترك مع
أنظمة غير لينكس

د. جُسِـــپ جُبرا إستيڤَ

مقدمة

بعد أن أخذنا مقدمة موجزة عن أنظمة جنو/لينكس، الخطوة التالية هي دمج هذه الأنظمة في بيئة العمل كأنظمة إنتاجية. اعتماداً على النظام المستخدم حالياً، يمكننا أن نبحث في الهجرة الكاملة إلى أنظمة جنو/لينكس، أو التواجد المشترك عبر خدمات متوافقة. يمكن الهجرة إلى أنظمة جنو/لينكس باستمرار وبالتدريج عبر استبدال الخدمات بشكل جزئي، أو استبدال كل ما هو موجود في النظام بما يكافئه في جنو/لينكس.

في بيئات التوزيعات الحالية، الاهتمام الأكبر في بيئات الخوادم/العملاء. أي مهمة في النظام الرئيسي تدار بخوادم مختصة، مع وجود تطبيقات أو مستخدمين بوصول مباشر إلى الخدمات المقدَّمة.

فيما يخص بيئة العمل، سواء كانت في أبسط حالاتها، كوجود مستخدم واحد، أو في الحالات الاكثر تعقيداً، كما في بيئات الأعمال، فستحتاج كل بيئة لمجموعة من الخدمات التي نحتاج لاختيارها ومن ثم ضبط أجهزة الخوادم والعملاء لتتمكن من تقديم أو الوصول إلى الخدمات.

قد تضم الخدمات مجالات مختلفة، ومن المعتاد وجود العديد من الأنواع لمشاركة الموارد والمعلومات. فمن الأنواع الشائعة خوادم الملفات، وخوادم الطباعة، وخوادم الوِب وخوادم الأسماء، وخوادم البريد، وغيرها.

سيختار المدير عادة مجموعة من الخدمات التي تحتاج بيئة العمل لوجودها، اعتماداً على حاجات المستخدمين النهائيّ و/أو المؤسسة، ويجب ضبط الدعم الصحيح للبنية التحتية، في نموذج للخوادم التي تدعم الحمل المتوقع.

1 أنظمة الحاسوب: البيئات

خلال عملية تثبيت بعض توزيعات جنو/لينكس، عادة ما نجد أننا نُسأل عن نوع البيئة أو المهام التي سيكون نظامنا مكرّساً لها، والتي عادة ما تتيح لنا اختيار مجموعات فرعية من البرمجيات التي ستُثبَّت لنا مبدئيًّا، وذلك لأنها الأكثر ملاءمة لتلك الوظيفة. سنُسأل عادة ما إذا كان النظام سيُستخدَم:

1) كمحطة عمل: هذا النوع من النظام يتضمن بالأساس تطبيقات معينة ستكون أكثر استخداماً. النظام موجه بالأساس لتشغيل هذه التطبيقات وعدد قليل من خدمات الشبكة.

2) خادم: يتضمن بالأساس معظم خدمات الشبكة، أو – في بعض الأحيان – خدمة معينة ستكون الخدمة الرئيسية للنظام.

3) وحدة حسابات متخصصة: تطبيقات حسابات مكثفة، وبرامج تصيير render، وتطبيقات علمية، وبرامج رسم حاسوبي CAD، وغيرها.

4) محطة رسومية: مكتبية مع تطبيقات تتطلب تفاعلاً رسومياً مع المستخدم.

يمكننا عادة ضبط نظام جنو/لينكس لدينا بواحد أو أكثر من هذه الخيارات.

> وبشكل أكثر عمومية، إذا كان علينا فصل بيئات العمل التي يمكن أن يستخدم فيها نظام جنو/لينكس، فيمكننا تحديد ثلاثة أنواع رئيسيّة لهذه البيئات: محطات العمل، الخوادم، والاستخدام المكتبي.

يمكننا أيضاً تضمين أنواع أخرى للأنظمة، والتي نطلق عليها عامة أجهزة "مدمجة" أو "مُضَمَّنة"، أو أنظمة نقّالة، كأجهزة المساعدات الشخصية، والهواتف النقالة، ومنصات الفيديو المحمولة، وغيرها. يوفر جنو/لينكس دعماً لهذه الأجهزة أيضاً، مع نواة صغيرة مخصصة لها.

مثال

على سبيل المثال، علينا ذكر العمل الابتدائي الذي قامت به شركة Sharp على نموذجها Zaurus، وهو مساعد

شخصي بمزايا لينكس متقدمة (هناك أربعة أو خمسة نماذج في السوق). أو مبادرات لينكس أخرى أيضاً، للأنظمة المضمنة مثل طرفيات نقاط البيع (POS). أو منصات الفيديو مثل GP2X، ودعم لينكس لمنصة الألعاب Playstation من شركة Sony. أضف إلى هذا، منصات الهواتف الذكية والمساعدات الشخصية الجديدة، مثل أندرويد من جوجل، وميمو من جوجل، و moblin من إنتل.

فيما يتعلق بالبيئات الرئيسية الثلاث، لنلقِ نظرة على كيفية تطور كل واحد من أنظمة الحاسوب هذه في بيئات جنو/لينكس.

1) النظام من نوع محطة العمل يكون جهازاً عالي الأداء ومستخدم لمهمة محددة، بدلاً من استخدامه لعدد من المهام العامة. تتكون محطات العمل التقليدية من جهاز عالي الأداء بعتاد محدد مناسب للمهمة التي نحتاج منه القيام بها؛ كانت في العادة Sun SPARC، أو IBM RISC، أو جهاز Silicon Graphics (وغيرها)، مع تنويعات من يونكس المملوك. كانت هذه الأجهزة المكلفة موجهة لقطاع معين من التطبيقات، سواء التصميم ثلاثي الأبعاد (في حالة Silicon و Sun)، أو قواعد البيانات (IBM أو Sun). أما الآن، فأداء العديد من الأجهزة الشخصية يقترب من (لكن لا يكافئ) هذه الأنظمة، والحد الفاصل بين هذه الأنظمة وجهاز شخصي لم يعد واضحاً، بفضل وجود جنو/لينكس كبديل لإصدارات يونكس المملوكة.

2) النظام من نوع الخادم له هدف محدد، وهو توفير خدمات للأجهزة الأخرى على الشبكة، و يوفر عدداً من الخصائص والوظائف التي يمكن تمييزها بوضوح عن الأجهزة الأخرى. في الأنظمة الحاسوبية الصغيرة (بأقل من 10 أجهزة مثلاً)، ليس هناك عادة نظام خادم حصري، وعادة ما تكون مشتركة مع وظائف أخرى، كنظام من النوع المكتبي مثلاً. الأنظمة المتوسطة (بضع عشرات من الأجهزة) فيها جهاز واحد أو أكثر مخصص للعمل كخادم، سواء كان جهازاً وحيداً يجعل كل الخدمات (بريد، ويب، إلخ) مركزية، أو زوجاً من الأجهزة المخصصة للتشارك في تقديم الخدمات الرئيسية.

في الأنظمة الكبيرة (مئات، بل وأحياناً آلاف الأجهزة)، يجعل الحملُ وجودَ مجموعة كبيرة من الخوادم ضرورياً، حيث يكون كلٌّ خادم عادة مخصّصًا لخدمة معينة، بل وحتى مجموعة من الخوادم المخصصة فقط لعملية واحدة. إضافة إلى هذا، إذا كانت هذه الخدمات تقدم لداخل أو لخارج المؤسسة، بالوصول عبر عملاء مباشرين أو عبر الإنترنت، اعتماداً على الحمل المعدة لدعمه، فسيكون علينا اللجوء إلى الحلول من النوع متعدد الأنوية SMP (تجميع

الأجهزة لتوزيع حمل خدمة معينة).

الخدمات التي قد نحتاجها داخلياً (أو خارجياً) يمكن أن تشمل التصنيفات التالية من الخدمات (وغيرها) :

- التطبيقات: يمكن للخوادم تشغيل تطبيقات، وكعملاء يمكننا فقط مراقبة تنفيذها والتفاعل معها. فمثلاً، يمكن أن تشمل خدمات طرفيات وتطبيقات تعمل عبر الوِب.
- الملفات: تقدم لنا مساحة مشتركة ويمكن الوصول إليها من أي مكان من الشبكة حيث يمكننا تخزين واستعادة ملفاتنا.
- قواعد البيانات: تجميع البيانات مركزياً للتشاور أو الإنتاج عبر تطبيقات النظام على الشبكة (أو الخدمات أخرى).
- الطباعة: هناك مجموعات من الطابعات، وطوابيرُها والوظائفُ المرسلة إليها من أي مكان على الشبكة مُدارة.
- البريد الإلكتروني: يقدّم خدمات استقبال وإرسال وإعادة إرسال الرسائل الواردة أو الصادرة.
- الوِب: خادم (أو خوادم) تملكها المؤسسة ليستخدمها الزبائن داخلياً أو خارجياً.
- معلومات الشبكة: من الضروري للمؤسسات الكبيرة إيجاد الخدمات المقدمة أو الموارد المشتركة؛ أو للمستخدمين أنفسهم، إذا كانوا يحتاجون خدمات تجعل هذا التوطين ممكناً وللعودة إلى خصائص كل نوع من الكيانات.
- خدمات الأسماء: مطلوب من الخدمات تسمية وترجمة الأسماء المختلفة التي يُعرَف بها المورِد الواحد.
- خدمات الوصول عن بعد: في حالة عدم امتلاك وصول مباشر، نحتاج إلى أساليب بديلة تسمح لنا

بالتفاعل مع النظام من الخارج، وتوفر لنا الوصول إلى النظام الذي نريده.

- خدمات إنشاء الأسماء: في تسمية الأجهزة على سبيل المثال، يمكن أن يكون هناك عدد كبير من التسميات، أو يمكن أن لا تكون كلها بنفس الأسماء دائماً. نحتاج لتوفير أساليب لتحديد هوية الأجهزة بدقة.

- خدمات الوصول للإنترنت: العديد من المؤسسات ليس لديها سبب للوصول إلى الإنترنت، أو في الغالب يكون لديها وصول عبر عبّارات أو خوادم وسيطة.

- خدمات الترشيح: إجراءات أمنية لترشيح [1] المعلومات غير الصحيحة أو التي تؤثر على الأمن.

3) الأجهزة من النوع المكتبي تكون ببساطة أجهزة مستخدمة للمهام الحاسوبية اليومية الروتينية (كالأجهزة الشخصية في المنزل أو المكتب).

مثال

على سبيل المثال، يمكننا أن نعتبر ما يلي مهام شائعة (مشمولة في بعض أكثر تطبيقات جنو/لينكس استخداماً):

- المهام المكتبية: توفير برمجيات تقليدية من حزمة المكتب: معالج النصوص، والجداول، والعروض التقديمية، وقواعد البيانات الصغيرة، وغيرها. يمكننا أن نجد حزماً مثل المكتب المفتوح، والمكتب الحرّ، و Calligra (المشتق عن Koffice من كدي)، وبرامج عديدة مثل Gnumeric و AbiWord الذين يمكن أن يشكلا جزءًا من حزمة لجنوم (تعرف أيضاً بمكتب جنوم).
- متصفّح وب: المتصفحات مثل فَيرفُكس، و konqueror، و epiphany، وغيرها.
- دعم العتاد (USB، وأجهزة التخزين ...). مدعومة في لينكس بالمشغلات المناسبة، والمتوفرة عادة في النواة أو من المُصنِّعين. هناك أيضاً أدوات لتحليل العتاد الجديد، مثل udev. الوسائط والتسلية (رسوميات، معالجة صور، تصوير رقمي، ألعاب، وغيرها). في جنو/لينكس كمّ هائل من هذه البرمجيات بجودة احترافية جدًّا: جمب Gimp (لمعالجة الصور)، Sodipodi, Xine, Mplayer, gphoto، وغيرها.
- الموصلية (الوصول لسطح المكتب البعيد، والوصول إلى الأنظمة الاخرى). في هذه الناحية، لجنو/لينكس كمّ هائل من الأدوات الخاصة به، سواء كانت TCP/IP أو FTP أو telnet أو وب أو غيرها، أو X Window الذي فيه توافقية مع سطح المكتب البعيد لأيّ جهاز يونكس، أو rdesktop (للاتصال بأجهزة وندوز)، أو VNC (للاتصال بيونكس أو وندوز أو ماك أو غيرها).

---

1 ترشيح: تصفية أو فَلتَرة filtering.

2 خدمات جنو/لينكس

لجنو/لينكس خوادم مُطوَّعة لأيّ بيئة عمل.

تصنيفات الخدمات التي ذكرناها لها مكافئات يمكننا توفيرها من أنظمة جنو/لينكس لدينا لكل الأجهزة الأخرى على الشبكة (والتي يمكنها أن تتعامل معها كعملاء أيضاً):

1) التطبيقات: يمكن لجنو/لينكس أن يوفر خدمات الطرفية البعيدة، سواء عبر الاتصال المباشر عبر سلسلة من الواجهات أو الطرفيات البسيطة التي تخدم في تمثيل المخرجات، أو عبر التفاعل مع التطبيقات. وهناك إمكانية أخرى، وهي الاتصال عن بعد في الوضع النصّيّ، من جهاز آخر عبر خدمات TCP/IP، مثل rlogin أو telnet أو بشكل آمن عبر ssh. يوفر جنو/لينكس خوادم لكل هذه الموافيق. في حالة تشغيل تطبيقات رسومية، لدينا حلول عبر X Window، حيث يمكن لأي يونكس أو لينكس أو وندوز رؤية البيئة العاملة وتطبيقاتها. وفي نفس الوقت، تتوفر حلول أخرى، مثل VNC لنفس الهدف. وفيما يخص تطبيقات الوِب، لجنو/لينكس خادم أباتشي، وأيّ من أنظمة وّب العديدة التي تعمل، سواء Servlets (مع Tomcat)، أو JSP, PERL, PHP, xml, webservices وغيرها. إضافة إلى خوادم تطبيقات وِب، مثل BEA Weblogic و IBM Websphere و Jboss (مجاني) والتي تعمل أيضاً على منصات جنو/لينكس.

2) الملفات: يمكن تقديم الملفات بطرق عديدة، سواء بالوصول إليها عبر FTP، أو بتقديمها بطريقة شفافة إلى أجهزة يونكس ولينكس عبر NFS، أو بالتصرف كعميل أو خادم مع أجهزة وندوز عبر سامبا.

3) قواعد البيانات: يدعم جنو/لينكس عدداً كبيراً من قواعد البيانات من النوع العلائقي للعميل/الخادم، مثل MySQL و PostgreSQL، وبعض قواعد البيانات المملوكة من هذا النوع، مثل Oracle و IBM DB2K والعديد غيرها.

4) الطباعة: يمكن أن يكون خادماً للطباعة لأنظمة يونكس عبر ميفاق TCP/IP، ولوِندوز عبر سامبا/CIFS.

5) البريد الإلكتروني: يوفر خدمت للعملاء للحصول على البريد على أجهزتهم (خوادم POP3 و IMAP)، كوكلاء نقل البريد (MTA) للحصول على البريد وإرساله، مثل خادم Sendmail (معيار يونكس)، أو غيره مثل Exim، أو خدمة SMTP للبريد الوارد في حالة الإرسال للخارج.

6) الوِب: لدينا خادم http أباتشي، [وخدمته في التوزيعات الردهاتية تسمى httpd]. يمكننا أيضاً تضمين خوادم تطبيقات وِب مثل Tomcat لـ servlets، و JSP ....

7) معلومات الشبكة: تسمح لنا خدمات مثل NIS و NIS+ و LDAP بجعل معلومات الأجهزة والمستخدمين والعديد من الموارد على شبكتنا مركزية، مما يسهل الإدارة وخدمة المستخدمين، وبهذه الطريقة لا تعتمد هذه الخدمات على حالتها على الشبكة. أو إذا كانت للمؤسسة هيكلية داخلية معينة، فستسمح لنا هذه الهيكلية بتشكيلها مما يسمح بالوصول إلى الموارد لمن يحتاجها.

8) خدمات الأسماء: الخدمات مثل DNS لأسماء الأجهزة وترجمتها من وإلى عناوين IP، باستخدام خادم Bind مثلاً (خادم الأسماء المعياري ليونكس).

9) خدمات الوصول البعيد: سواء لتشغيل التطبيقات أو للحصول على معلومات بعيدة عن الأجهزة. يمكن أن تكون الخوادم هي ما ذكرناه للتطبيقات: Xwindow و VNC وغيرها، وتلك التي تسمح بتنفيذ بعض الأوامر البعيدة دون تفاعل أيضاً، مثل rexec, rsh, ssh، وغيرها.

10) خدمات إنشاء الأسماء: خدمات مثل DHCP تسمح لشبكات TCP/IP أن تُنشئ عناوين IP المتاحة بشكل متغير (أو ثابت) اعتماداً على الأجهزة التي تحتاجها.

11) خدمات الوصول للإنترنت: يمكن في حالات معينة أن يكون هناك مخرج واحد للإنترنت (أو أكثر). تعمل هذه النقاط كوسيط، حيث لديها وصول للإنترنت، وتقوم بإعادة توجيه الوصول المحتمل إلى الإنترنت من العملاء. وعادة ما تعمل أيضاً تخزيناً مؤقتاً للمحتوى. يمكننا في جنو/لينكس مثلاً الحصول على سكويد Squid. في هذا التصنيف، يمكن أن يستخدم جنو/لينكس أيضاً كعبّارة أو موجِّه، سواء لإعادة توجيه الحزم إلى شبكات أخرى،

أو لإيجاد مسارات بديلة لإعادة الإرسال. يمكننا أيضاً في حالة التثبيتات الصغيرة كالمنزلية أن نضمّن الوصول إلى الإنترنت عبر المودِم باستخدام خدمات PPP.

12) خدمات الترشيح: جدران الحماية من أكثر الإجراءات الأمنية استخداماً في الوقت الحاضر. تمثل جدران الحماية ببساطة تقنيات ترشيح للحزم الواردة والصادرة، للموافيق المختلفة التي نستخدمها، ولوضع حدود للحزم غير المرغوبة. في جنو/لينكس لدينا آليات مثل ipchains و iptables (الأحدث) لتنفيذ جدران الحماية.

3 أنواع الاستخدام

يوفر نظام جنو/لينكس خصائص صالحة للاستخدام الشخصي، ولمستخدمي البُنى التحتية المتوسطة والكبيرة أيضاً.

من وجهة نظر مستخدمي نظام جنو/لينكس، علينا أن نميز:

1) المستخدم الفرد أو المنزلي: عادة ما يكون لدى المستخدم من هذا النوع جهازاً واحداً أو أكثر في المنزل، والتي قد تكون مشتركة او غير مشتركة. وبشكل عام، يُستخدَم جنو/لينكس في هذه البيئات لتطوير نظام مكتبي، مما يعني أن وجود جزء رسومي سيكون مهماً: جنو/لينكس المكتبي. لدينا لهذا الجهاز المكتبي خياران رئيسيان وهما بيئتا جنوم وكدي، وكلتاهما صالحتان تماماً. توفّر كلّ من البيئتين تطبيقات تعمل وخدمات للرسوميّات، إضافة إلى نطاق عريض من التطبيقات البسيطة للمستخدم والتي تسمح لنا بتطوير كل أنواع المهام اليومية. توفر كلتا البيئتان سطح مكتب رسومي، مع قوائم مختلفة، أشرطة أيقونات وأيقونات، إضافة إلى متصفحات للملفات الشخصية وتطبيقات عديدة مفيدة. يمكن لأيّ بيئة تشغيل تطبيقاتها الخاصة وتطبيقات أخرى، ولكن كما هي حال التطبيقات، فهي تعمل بشكل أفضل على بيئاتها الخاصة، وذلك لأن النواحي المرئية مناسبة أكثر للبيئات التي طوِّرَت لها. فيما يخص تطبيقات الاستخدام الشخصي، علينا تضمين التطبيقات الاعتيادية للنظام المكتبي. إذا كان لدى المستخدم شبكة منزلية، كوجود مجموعة صغيرة من الحواسيب المرتبطة مع بعضها عبر شبكة سلكية Ethernet، يمكن أن تكون خدمات مشاركة الملفات والطابعات بين الأجهزة مثيرة للاهتمام. قد تكون الخدمات من أمثال NFS ضرورية إذا كان هناك أجهزة لينكس أخرى؛ أو سامبا إذا كان هناك أجهزة وِندوز.

في حال وجود اتصال بالإنترنت عبر مزوِّد خدمة إنترنت ISP، واعتماداً على نوع الاتصال المستخدَم، يمكن ان نحتاج للتحكم بالأجهزة والموافيق المرتبطة:

- اتصال مودِم: تستخدم مودِمات الهاتف ميفاق PPP للاتصال بالمزوِّد. سيكون علينا أن تفعيل هذا الميفاق وضبط الحساب الذي فعله لنا المزود. المشكلة الهامة مع لينكس هي قضية مودِمات winModems، والتي تسبب كثيراً من المشاكل. هذا المودِم ليس مدعوماً (مع بعض

الاستثناءات)، وذلك لأنه ليس مودِماً حقيقيًّا، بل تبسيط للعتاد مع برمجية مشغِّل، ومعظمها تعمل فقط على وِندوز، مما يعني أن علينا تجنبها (إذا لم تكن مدعومة) وشراء مودِمات حقيقيّة (كاملة).

- اتصالات مودِمات ADSL: سيكون العمل مشابهاً، فيمكن استخدام ميفاق PPP أو واحداً آخر اسمه EoPPP. قد يعتمد هذا على مصنّع المودم، وعلى نوع المودم: إثرنت أو USB.

- اتصال ADSL مع موجه: الضبط سهل جدًّا، لأن كل ما سيكون علينا فعله في هذه الحالة ضبطُ بطاقة الشبكة السلكيّة أو اللاسلكية في نظامنا والاتصال بموجه ADSL.

ما إن تصير الواجهة للإنترنت متصلة ومضبوطة، فآخر نقطة هي تضمين نوع الخدمات التي سنحتاجها. إذا كنا نريد فقط التصرف كعميل على الإنترنت، فمن المجدي استخدام أدوات عميلة للموافيق المختلفة، سواء FTP، أو telnet، أو البريد الإلكتروني، أو قارئات الأخبار، وغيرها. إذا كنا نرغب أيضاً بتوفير خدمات صادرة – كأن ننشر موقعاً (بخادم وِب) أو أن نسمح للوصول إلى جهازنا من الخارج (عبر خدمات ssh, telnet, FTP, X Window, VNC، إلخ)، وفي هذه الحالة يصير الجهاز خادماً – فعلينا أن نتذكر أن هذا سيكون ممكناً فقط إذا كان مزودنا يعطينا عناوين IP ثابتة لأجهزتنا. عدا هذا، فسيتغير عنواننا كل مرة نتصل بها، وإمكانية تقديم خدمة ستصير إما صعبة جدًّا أو مستحيلة[2].

ومن الخدمات الأخرى المثيرة للاهتمام مشاركة الوصول إلى الإنترنت بين أجهزتنا المتاحة.

2) المستخدم المتوسط: هذا المستخدم في المؤسسات متوسطة الحجم، سواء شركة صغيرة أو مجموعة مستخدمين. في العادة، سيكون لهذا النوع من المستخدمين اتصالٌ بالشبكة المحلية (عبر شبكة LAN مثلاً) مع بعض الأجهزة والطابعات المتصلة. وسيكون لديه وصولٌ مباشر للإنترنت، سواء عبر وسيط ما (نقطة أو جهاز مصمم لاتصال خارجي)، أو سيكون هناك أجهزة قليلة متصلة فيزيائيًّا بالإنترنت. بشكل عام، في هذه البيئة، العمل بعضه محلي

---

2 يمكن الالتفاف على المشكلة باستخدام ما يعرف بخدمة أسماء النطاقات المتغيرة Dynamic DNS، وهناك مواقع توفر نطاقات فرعية مجانية. سنحتاج أيضاً لضبط جدار الحماية والموجه لتمرير الاتصالات الواردة إلى الخادم.

وبعضه مشترك(سواء موارد أو طابعات أو تطبيقات). سنحتاج عادة أنظمة مكتبية، فمثلاً، يمكننا في المكتب استخدام تطبيقات الحزم المكتبية إضافة إلى عملاء إنترنت، وربما أنظمة من نوع محطات العمل أيضاً، مثلاً، لأعمال الهندسة، والمجال العلمي، ويمكن استخدام تطبيقات الرسم الحاسوبي أو تطبيقات معالجة الصور، إضافة إلى أنظمة الحسابات الرياضية المكثفة، وغيرها، وربما من المؤكد وجود أجهزة قوية يتم توكيلها بهذه المهام.

في بيئة المستخدم هذه، سيكون علينا عادة أن نشارك الموارد، مثل الملفات، والطابعات، وربما التطبيقات، وغيرها. ولهذا، ستكون خدمات NFS مناسبة في نظام جنو/لينكس، وكذلك خدمات الطباعة، وسامبا (إذا كان هناك أجهزة وندوز نحتاج لمشاركة الملفات والطابعات معها)، وسنحتاج أيضاً بيئات قواعد بيانات، خادم وِب داخلي مع تطبيقات مشتركة، وغيرها.

3) مستخدمو البيئات الكبيرة: يشبه هذا النوع سابقه ويختلف فقط في حجم المؤسسة والموارد المتاحة، والتي يمكن أن تكون كثيرة، وفي حالة كهذه يمكن أن نحتاج بعض موارد NIS أو +NIS أو LDAP للتحكم بمعلومات المؤسسة، وتوفير صورة عن هيكليتها، وبالتأكيد امتلاك بنية تحتية ضخمة للخدمات للمستخدين الخارجين أيضاً، وعادة ما تكون بهيئة مواقع مع تطبيقات مختلفة.

لهذا النوع من المؤسسات مستوى عالٍ من التنوع في كلّ من عتاد وبرمجيات النظام، ويمكننا أن نجد العديد من المعماريات وأنظمة التشغيل المختلفة، مما يعني أن المهام الرئيسية ستتكون من تسهيل توافقية البيانات، وذلك بالاعتماد على موافيق وعملاء وخوادم معيارية (عادة مع عناصر TCP/IP).

4 الهجرة أو التواجد المشترك

فيما يلي، سنأخذ ناحية أخرى مهمة في تبني أنظمة جنو/لينكس بعين الاعتبار. لنفترض بأننا هواة في التعامل مع هذا النظام، أو العكس، أننا خبراء ونرغب بتبني نظام جنو/لينكس واحد أو أكثر كمستخدمين فرديين للعمل في مؤسستنا الصغيرة، أو أننا نفكر جدّيًّا باستبدال البنية التحتية لمؤسستنا أو شركتنا الكبيرة بشكل كامل (أو جزئيًّا).

الهجرة إلى نظام جديد ليس أمراً بسيطاً، بل تحتاج إلى تقييم بدراسة تحلل التكاليف والمزايا المستفادة التي نتوقع الحصول عليها من هذه الهجرة. وأيضاً، يمكن أن تتم الهجرة بشكل كامل أو جزئيّ، بدرجة معينة من التواجد المشترك مع الأنظمة القديمة.

سنتعامل مع مشروع هجرة كامل أو جزئي لنظام تقنية المعلومات لدينا إلى جنو/لينكس، وسنكون - كمدراء للنظام - مسؤولين عن هذه العملية.

كما في أيّ مشروع، سيكون علينا أن ندرس طريقة الإجابة على أسئلة مثل: هل ستكون الهجرة منطقية فيما يتعلق بالمنفعة المادية أو الأداء؟ ما الهدف من الهجرة؟ ما المتطلبات التي نريدها أو نحتاج لتنفيذها؟ هل سنحتاج لعمل هجرة جزئية أم كليّة؟ هل التواجد المشترك مع أنظمة أخرى ضروريّ؟ هل سيكون علينا الاحتفاظ بالمستخدمين؟ هل سيكون بإمكاننا استخدام نفس العتاد؟ أم سنحتاج لعتاد جديد؟ هل سيكون هناك تكاليف إضافيّة هامّة؟ أو ببساطة، هل ستسير الأمور على ما يرام؟ هذه الأسئلة وغيرها الكثير سنحتاج لمحاولة إجابتها. في حالة الشركات، يمكن توفير الإجابة كمشروع هجرة، بتحديد الأهداف، والمتطلبات، وعملية التنفيذ، وتضمين تحليل ماليّ، وخطط تدريب المستخدمين، وغيرها. لن ندخل في هذا بالتفصيل، ولكننا سنتعامل مع بعض هذه القضايا بشكل بسيط. وفي ورشة العمل النهائية، سنختبر بعض الحالات الصغيرة لكيفية تنفيذ الهجرة.

إضافة إلى هذا، في لحظة البدء بالهجرة إلى جنو/لينكس، سنبدأ بملاحظة المزايا التي يجلبها النظام لمؤسستنا:

1) التكاليف: تقليل تكاليف الترخيص لتطبيقات وبرمجيات النظام. تكلفة ترخيص جنو/لينكس هي صفر إذا تم الحصول عليه من الإنترنت (بصيغة صورة لقرص التوزيعة مثلاً)، أو بتكلفة لا تذكر إذا ما أخذنا بعين الاعتبار أن أقرب مقارنة للأنظمة ذات المزايا المشابهة ستكون مع أنظمة وِندوز للخوادم مع تكلفة ترخيص تتراوح ما بين

1500 و 3000 يورو، هذا إذا تجاهلنا الكم الهائل من البرمجيات الإضافية التي تكون مضمّنة في أيّ توزيعة جنو/لينكس اعتيادية.

لكن احذر، فعليك أن لا تستخف بتكاليف المتابعة والتدريب. إذا كانت مؤسستنا تتكون فقط من مستخدمين ومدراء مدربين على وندوز، قد تكون تكاليف تدريب الطاقم كبيرة، وربما تكاليف المتابعة أيضاً. ولهذا تفضّل العديد من الشركات الاعتماد على توزيعات جنو/لينكس التجارية لتنفيذ ومتابعة النظام، كالإصدارات التجارية التي توفرها ردهات وسوزي وغيرها. لهذه الإصدارات من جنو/لينكس تكاليف ترخيص مرتفعة (مقارنة بوندوز)، ولكن – في نفس الوقت – هذه التوزيعات مهيئة لطبيعة العمل، وتحوي برمجياتها الخاصة لإدارة البنى التحتية لتقنية المعلومات في الشركة. ومن النواحي الأخرى الهامة أيضاً والتي علينا استنتاجها في تقدير التكاليف موضوع التكلفة الكلية للملكية Total Cost of Ownership – TCO، كتقييم عام للتكاليف اللازمة التي سنواجهها عندما نهمّ بعمل تطوير تقنيّ، علينا أن لا نقيّم تكاليف التراخيص والأجهزة فقط، بل وتكاليف التدريب والدعم للمستخدمين والتطبيقات المعتمدة أيضاً، والتي قد تكون مثل أو أكثر من الحلّ المُنَفَّذ.

2) الدعم: يقدم جنو/لينكس أفضل دعم للصيانة بين كلّ الأنظمة القديمة والحديثة، وغالباً ما تكون مجانية. رغم هذا، فبعض الشركات تمانع تبني جنو/لينكس على أساس عدم وجود دعم للمنتَج، وتفضّل شراء توزيعات تجارية تأتي مع عقود صيانة. لجنو/لينكس مجتمع دعم قويّ حولى العالم، عبر مؤسسات عديدة توفر توثيقاً مجانيًّا (الأدلة Howto المعروفة)، منتديات متخصصة للمستخدمين، مجتمعات للمستخدمين في كل قطر أو بلد في العالم تقريباً، إلخ. يمكن البحث عن أيّ سؤال أو مشكلة نواجهها على الإنترنت، ويمكننا إيجاد إجابات خلال دقائق. إذا لم نجد، أو إذا وجدنا علّةً أو خطأً أو موقفاً لم يتم اختباره، فيمكننا الإبلاغ عنه على العديد من المواقع (منتديات، مواقع تطوير، مواقع علل التوزيعات، وغيرها)، وأن نحصل على حلول خلال ساعات، أو على الأكثر خلال أيام. في أيّ وقت تكون لدينا فيه مشكلة أو سؤال، فعلينا أولاً تجربة بعض الإجراءات (وهكذا سنتعلم)، وإذا لم نجد الحلّ خلال وقت معقول، فسيكون علينا استشارة مجتمع جنو/لينكس في حالة إذا كان أيّ مستخدم أو مجموعة من المستخدمين قد واجهوا نفس المشكلة ووجدوا حلًّا، وإن لم يكن، فيمكننا دائماً إرسال تقرير عن المشكلة

ونرى إذا قدِّمت لنا حلول.

4.1 تحديد متطلبات الخدمة

في العادة، إذا كان لدينا أنظمة تعمل مسبقاً، فسيكون علينا إيجاد خدمات منفّذة للمستخدمين أو لمساعدة البنية التحتية لخدمة تقنية المعلومات. ستقع الخدمات ضمن بعض الفئات المذكورة أعلاه، مع خيارات جنو/لينكس التي ذكرناها.

أنظمة جنو/لينكس ليست جديدة على الإطلاق، وكما رأينا في المقدمة، فهي متفرعة عن تاريخ يمتد لأكثر من أربعين عاماً من استخدام وتطوير أنظمة يونكس. ولهذا، من أول الأمور التي سنجدها هي أننا لا نفتقر للدعم لأيّ نوع من الخدمة التي نريدها. إذا كان هناك من شيء، فهو الاختلاف في طريقة عمل الأشياء. وأيضاً، العديد من الخدمات التي تستخدمها أنظمة تقنية المعلومات قد تم تصوّرها، وبحثها، وتطويرها، وتنفيذها على يونكس من يومها، وتم تطويعها لاحقاً للأنظمة الأخرى (مثل ويندوز، وليس بنفس مستوى النجاح).

> كثير من الشركات التي لديها يونكس مملوك تساهم في جنو/لينكس وتقدم بعض تطويراتها للمجتمع.

أي خدمة متوفرة حاليًّا يمكن تبنيها في أنظمة جنو/لينكس بخدمات بديلة (إن لم يكن نفسها).

مثال

من الحالات المشهورة خوادم سامبا. يقدم وندوز ما يسميه "مشاركة الملفات والطباعة على الشبكة" عبر ميفاق خاص بها يعرف بشكل عام بمسمى server message block – SMB (مع دعم للشبكة عبر موافيق NetBios و NetBEUI). الاسم CIFS (أو نظام ملفات الإنترنت الشائع common Internet file system) يعتبر أيضاً شائع الاستخدام، وهو ما سمي الإصدار الثاني اعتماداً عليه (والذي بقي يستخدم SMB كميفاق أساسيّ). تسمح هذه الموافيق بمشاركة الملفات (أو الأقراص) والطابعات على شبكة لأجهزة وندوز (في إعداد مجموعة عمل أو نطاق وندوز). لقد كانت هذه الفكرة قديمة في يونكس عندما ظهرت في وندوز، وكانت هناك خدمات مثل NFS لمشاركة الملفات، وإدارة الطابعات عن بعد كانت متوفرة بالفعل عبر موافيق TCP/IP.

من المشاكل المتعلقة باستبدال خدمات مشاركة الملفات لوندوز والمبنية على NetBios و NetBEUI (وبالكامل عبر NetBios على TCP/IP) كنت متعلقة بكيفية دعم هذه الموافيق، حيث إذا أردت الاحتفاظ بأجهزة عميلة تعمل بوندوز، فلا يمكننا استخدام خدمات يونكس. ولهذا الهدف تم تطوير سامبا كخادم يونكس يدعم موافيق وندوز ويمكنه استبدال جهاز خادم/عميل وندوز دون إحداث فرق، بحيث لا يحتاج مستخدمو عملاء وندوز لملاحظة شيء على الإطلاق. إضافة إلى هذا، فقد كانت النتائج في معظم الحالات مماثلة في الأداء إن لم تكن أفضل من الجهاز الأصلي بخدمات وندوز.

حالياً، يتطور سامبا باستمرار للإبقاء على التوافقية مع خدمات مشاركة الملفات والطباعة لوندوز؛ وذلك نتيجة للتغييرات الرئيسية التي تجريها مايكروسوفت على موافيق SMB/CIFS (الأساس موجود في سامبا) مع كل إصدار جديد من وندوز، وتحديداً التطور في تصاميم مجموعات العمل في إصدارات عملاء انظمة التشغيل، للحصول على خادم مركزي (أو مجموعة خوادم) لهذه التصاميم، بخدمات استيثاق مستخدمين معينة (NTLM, NTLMv2, Kerberos)، والتخزين المركزي لإدارة النظام، مثل Active Directory. إضافة إلى هذا، ضبط خوادم النطاق الموجودة (سواء بمتحكم رئيسي، أو نسخ احتياطي، أو Active Directory).

سنحتاج حاليًّا في عمليات الهجرة مع سامبا إلى التدقيق في إعدادات خوادم وعملاء وندوز الموجودة في النظام،

إضافة إلى استيثاق الستخدمين و/أو أنظمة إدارة المعلومات المستخدمة. وأيضًا، سنحتاج لمعرفة هيكلية النظام وتقسيمه إلى نطاقات (وخوادم التحكم، والأعضاء، والخوادم المعزولة)، وذلك لنتمكن من عمل خطة كاملة وصحيحة للحلول المبنية على سامبا، واستيثاق المستخدمين المستخدم لإكمال العمل (winbind, kerberos, nss_ldap) وخدمات الإدارة (مثل OpenLDAP).

4.2 عملية الهجرة

نحتاج في عملية الهجرة إلى أخذ طريقة الهجرة التي نريدها بعين الاعتبار، إذا كنا نريد الهجرة بالكامل أو جزئيًّا، مع تواجد مشترك مع خدمات أو معدات أخرى لها نظام تشغيل مختلف.

يمكننا ترحيل عناصر عديدة، سواء الخدمات التي نقدمها، أو الأجهزة التي تقدم الخدمات إلى العملاء الذي يصلون إلى هذه الخدمات.

العناصر التي يمكن ترحيلها تشمل:

1) الخدمات أو الأجهزة المتخصصة في خدمة واحدة أو أكثر. سنستبدل في الهجرة كلّ خدمة ببديل آخر مكافئ لها، عادة بأقل أثر ممكن، مالم نرِد تغيير العملاء أيضاً. في حالة عملاء وندوز، يمكننا استخدام خادم سامبا لاستبدال خدمات الملفات والطباعة التي توفرها أجهزة وندوز. والخدمات الأخرى، يمكننا تبديلها إلى مكافئاتها في جنو/لينكس. في حال استبدال خدمة واحدة فقط، فسنعطّل بالعادة هذه الخدمة على الجهاز الذي كان يقدمها ونفعّلها على النظام الجديد. قد تكون التغييرات على العملاء ضروريّة (كعناوين الجهاز الجديد مثلاً، أو المعاملات ذات العلاقة بالخدمة).

إذا كان جهاز خادم مسؤولاً عن عمل كامل، فسيكون علينا تحليل ما إذا كان الجهاز متخصصاً لخدمة واحدة أو أكثر، وإذا ما كان يمكن استبدال هذه الخدمات بالكامل. إذا كان ممكناً، فسيكون علينا فقط تبديل الجهاز الجديد مكان الجهاز القديم (أو الإبقاء على القديم)، حيث يحوي الجديد تلك الخدمات تحت جنو/لينكس، وفي أيّ حال، سنعدل متغيرات العملاء عند الضرورة. في عادة، قبل عمل تغيير، ينصح بختبار الجهاز بشكل منفصل، مع عملاء قلة، وذلك للتأكد من أنه يقوم بالعملية على الوجه الصحيح، ومن ثمّ تبديل الأجهزة خلال فترة خمول النظام.

وفي أيّ حال، سيكون علينا بالتأكيد أخذ نسخة احتياطية للبيانات الموجودة قبل النظام الجديد، مثل نظام الملفات أو التطبيقات المتاحة في الخادم الأصليّ. وهناك نقطة أخرى علينا أخذها بعين الاعتبار مسبقاً، وهي

محمولية[3] البيانات، من المشاكل التي نجدها كثيراً التوافقية عند استخدام المؤسسة لبيانات أو تطبيقات معتمدة على المنصة.

مثال

لنذكر بعض الحالات العملية التي تواجهها بعض الشركات هذه الأيام:

- تطبيقات الوب مع ASP: يمكن تنفيذ هذه التطبيقات على منصات وب وندوز وخادم الوب Microsoft IIS فقط. علينا تجنبها إذا كنا نرغب بتحويل المنصات أيّ وقت ولا نرغب بإعادة كتابتها أو الدفع لشركة أخرى للقيام بالأمر. لمنصات جنو/لينكس خادم وب أباتشي (الأكثر استخداماً على الإنترنت)، والذي يمكن استخدامه مع وندوز أيضاً، ويدعم هذا الخادم ASP في Perl (في وندوز يستخدم عادة Visual Basic و C# و Javascript)، وهناك حلول من أطراف خارجية لنقل ASP أو ربما تحويلها. لكن إذا اعتمدت شركتنا على هذا، فسيكون هذا مكلفاً في ما يخص الوقت والمال. من الممكن القيام بعملية التطوير بلغة جافا كحل عملي (المحمولة بين المنصات)، أو حلول أخرى مثل PHP. في هذه النقطة، علينا التنبيه على مشروع مونو Mono (مدعوم من نوفِل) الذي يجعل جزءًا من بيئة NET. محمولة إلى جنو/لينكس، وتحديداً عدد هائل من واجهات برمجة التطبيقات API لـ NET.، ولغة C#، ومعيار ASP.NET. مما يسمح بهجرة مرنة لتطبيقات NET. المبنية على واجهات برمجة التطبيقات التي تدعمها منصة مونو. وفي نفس الوقت، علينا أن نذكر شروع DotGNU من FSF، كبديل برخصة GPL لمونو.
- قواعد البيانات: استخدام خادم Microsoft SQL Server مثلاً يجعلنا معتمدين كليًّا على منصة وندوز، وأيضاً، إذا كنا نستخدم حلول مملوكة في بيئة لقواعد البيانات، فسيكون من الصعب نقلها. قواعد البيانات الاخرى مثل Oracle و DB2 (من IBM) محمولة أكثر لأن لها إصدار لمنصات مختلفة، أو لأنهم يستخدمون لغة برمجة محمولة أكثر. يمكننا أيضاً العمل مع أنظمة قواعد بيانات PostgreSQL أو MySQL (لها أيضاً إصدار لوندوز) المتوفرة في جنو/لينكس، وهذا يوفر انتقالاً أسهل. في نفس الوقت، إذا جمعناها مع تطوير وب فلدينا الكثير من الإمكانيات؛ وبهذا الخصوص، نستخدم هذه الأيام أنظمة مثل: تطبيقات الوب مع جافا، سواء servlets أو applets، أو EJP؛ أو حلول مثل LAMP المشهور، وهو تجميعة من جنو/لينكس، وأباتشي، و MySQL و PHP.

2) محطة العمل: تنبع المشكلة الرئيسية في الهجرة من التطبيقات، سواء كانت تطبيقات رسم حاسوبي، أو رسوم متحركة، أو برامج هندسية أو علمية، وهي السبب الرئيسي في وجود محطات العمل هذه. من المهم هنا أن يكون بالإمكان استبدالها بتطبيقات مساوية أو على الأقل متوافقة معها بنفس المزايا والوظائف المتوقعة. في العادة، معظم هذه التطبيقات تنبع من عالم يونكس، على اعتبار أن هذه المحطات صممت كأجهزة يونكس. هذا يعني أن تصريفها أو تطويعها بشكل طفيف لتناسب جنو/لينكس الجديد قد يكون كافياً، إذا كان لدينا المصدر البرمجي (كما هي العادة في حالة البرمجيات العلمية). إذا كنا نتعامل مع تطبيقات علمية، فالمصنعون (للبرمجيات الهندسية والعلمية) قد بدأوا بتطويعها لجنو/لينكس، لكن في هذه الحالة التطبيقات عادة تكون غالية جدًّا (مئات إلى آلاف الدولارات على الأقل).

---

3 المحمولية portability هي قابلية شيء ما (وهو في هذه الحالة البيانات) للنقل من مكان لآخر، وهي هنا إمكانية نقل البيانات من الخادم القديم غلى الجديد دون أضرار أو آثار جانبية.

3) أجهزة العملاء المكتبية: ما تزال الأجهزة المكتبية تسبب صداعاً لعالم جنو/لينكس، لأنها تتضمن مشاكل إضافية. في الخوادم، الأجهزة موكلة بالقيام بمهام واضحة، وهو دور لا تحتاج فيه إلى واجهات رسومية معقدة (عادة يكون التواصل النصي كافياً)، وعادة ما يشترى لها عتاد محدد وعالي الأداء للقيام بمجموعة معينة من المهام، والتطبيقات تكون في ذاتها خوادم مضمنة في نظام التشغيل أو تطبيقات أطراف خارجية. وأيضاً، عادة ما يدير هذه الأجهزة مدراء أنظمة ذوي معرفة واسعة بما يتعاملون معه. بينما في حالة الأجهزة المكتبية فنحن نتعامل مع عامِلٍ مشاكل (في ذاته، وأكثر من ذلك بالنسبة للمدراء): المستخدم النهائي للنظام. يتوقع مستخدمو الأنظمة المكتبية أن يحصلوا على واجهة رسومية قوية وسهلة نوعاً ما، وتطبيقات تسمح لهم بالقيام بالمهام الروتينية – عادة أعمال المكتب –. هذا النوع من المستخدمين (مع استثناءات قليلة) لا حاجة له بالحصول على معلومات متقدمة في الحاسوب؛ بشكل عام، هم معتادون على تطبيقات المكتب ويستخدمون بضع تطبيقات بمهارات متفاوتة. لجنو/لينكس هنا مشكلة واضحة، لأن يونكس لم يتخيل يوماً كنظام مكتبي صَرِف بهذا الشكل، ولم يطوّع لاستخدام واجهات رسومية إلا في وقت متأخر عبر X Window وأسطح المكتب المختلفة، كأسطح المكتب الحالية لأنظمة جنو/لينكس: جنوم وكدي. بل وأكثر من هذا، فالمستخدم النهائي معتاد على أنظمة وندوز (والتي لها حصة تعادل 95% من سوق الحواسيب المكتبية تقريباً).

في حالة الحواسيب المكتبية، لجنو/لينكس عدد من العوائق التي عليه تخطيها. من العوائق الحرجة التي يواجهها هي أنه لا يأتي مثبتاً مسبقاً على الأجهزة، مما يجبر المستخدمين على امتلاك قدر معين من المعرفة ليتمكنوا من تثبيته[4]. الأسباب الأخرى قد تشمل:

ملاحظة:

معركة البيئات المكتبية لأنظمة جنو/لينكس لم تشتعل بعد؛ وهي تحتاج للتغلب على امتناع المستخدمين عن تبديل الأنظمة وإنشاء وعي بقدرتهم على توفير بدائل وتطبيقات بسيطة يمكنها التعامل مع المهام التي يطلبها المستخدمون

- امتناع المستخدمين: قد يسأل المستخدم: لم علي تغيير النظام؟ هل ستوفر لي البيئة الجديدة نفس الأشياء؟ من الأسباب الأساسية للتغيير جودة البرمجيات وتكلفتها، حيث ستكون كمية كبيرة منها

4 هذا الأمر تغير قليلاً في السنوات الأخيرة، حيث صار عدد من مصنعي العتاد يوفر أنظمة لينكس مثبتة لتوزيعات مثل أبونتو وسوزة.

مجانية. في هذه النقطة، علينا أخذ قضية البرمجيات غير القانونية بعين الاعتبار. يبدو أن المستخدمين يعتبرون برمجياتهم مجانية، بينما هم في الحقيقة في وضع غير قانوني. تقدم برمجيات جنو/لينكس جودة عالية وتكلفة ضئيلة (أو دون تكلفة في حالات كثيرة)، مع بدائل عديدة لنفس الوظيفة.

- البساطة: يشعر المستخدمون عادة بالضياع إذا لم تكن في النظام نقاط مرجعية مشابهة لما هم معتادون عليه مسبقاً، كسلوك الواجهة أو أدوات بوظائف مشابهة. يتوقع المستخدمون عموماً أن لا يقضوا وقتاً طويلاً في تعلم كيفية التعامل مع النظام الجديد. ما زال لجنو/لينكس تقريباً مشاكل قليلة فيما يخص التثبيت التلقائي، مما يعني أنه ما زال يتطلب قدراً معيناً من المعرفة لتثبيته بشكل صحيح. في هذه النقطة، علينا أن نذكر سهولة تثبيته في بيئات عديدة توفرها التوزيعات الحديثة الموجهة للاستخدام المكتبي مثل أبونتو. وهناك مشكلة أخرى شائعة تتعلق بدعم عتاد الأجهزة الشخصية، ورغم أنه يتحسن من هذه الناحية مع مرور الوقت، إلا ان المصنعين ما زالوا لا يعيرونه اهتماماً كافياً (جزئيًا لأسباب تتعلق بالحصة السوقية). إلى أن يصبح هنالك نية واضحة بهذا الخصوص، فلن يكون بإمكاننا الحصول على نفس الدعم المتوفر للأنظمة الأخرى المملوكة (مثل وندوز). رغم هذا، فعلينا أن نبرز عمل مجتمع نواة لينكس الذي يقدم الدعم الصحيح للتقنيات الحديثة، بدعمهم للمصنعين في بعض الحالات، وفي تحضير دعم رئيسي (إذا لم يكن العتاد مدعوماً من المصنّع)، أو دعم بديل للذي يوفره المصنع.

- الشفافية: لبيئات جنو/لينكس العديد من الآليات المعقدة، كالمراقبات (العفاريت daemons)، والخدمات، وملفات ASCII صعبة الضبط، وغيرها. بالنسبة للمستخدمين النهائيين، من المفترض أن يكون إخفاء كل هذه التعقيدات ضرورياً، وذلك باستخدام برامج رسومية، معالجات ضبط، وغيرها. هذا هو المسار الذي اتخذته بعض التوزيعات مثل ردهات وماندريفا وأبونتو وغيرها.

- دعم التطبيقات المعروفة: سيواجه مستخدم حزم المكتب العادي مشكلة محمولية البيانات والتعامل مع هيئات البيانات. ما العمل مع البيانات الحالية؟ هذه المشكلة تحل باستمرار، وذلك بفضل حزم

المكتب التي صارت تحوي الإمكانيات التي يحتاجها المستخدم المكتبي. على سبيل المثال، إذا كنت تفكر بالهجرة من استخدام حزمة المكتب من مايكروسوفت، فيمكننا إيجاد حزم مثل المكتب المفتوح (برمجية حرة) يمكنها قراءة (وإنشاء) صيغ ملفات المكتب (مع بعض القيود). التوافقية في الصيغة لا تكون مشكلة إذا كانت مفتوحة، ولكن في حالة وندوز، فما تزال مايكروسوفت تحتفظ بسياسة الصيغ المغلقة؛ وهناك قدر كبير من العمل نحتاجه لنتمكن من استخدام هذه الصيغ، وذلك عن طريق الهندسة العكسية (عملية مكلفة). وأيضاً، في عصر الإنترنت، عندما يفترض أن تتحرك المعلومات بحرية، فإن الصيغ المغلقة غير الموثقة تشكل عائقاً أكثر من أيّ شيء آخر. الأفضل استخدام صيغ مفتوحة مثل RTF (رغم ان هذه لها مشاكل أيضاً بسبب عدد الإصدارات الكبير)، أو الصيغ المبنية على XML (ينشئ المكتب المفتوح ملفاته الخاصة بصيغة XML)، أو PDF للوثائق الموجهة للقراءة فقط. علينا أيضاً أن نشير إلى المجهود الذي بذله مجتمع المكتب المفتوح في إنشاء صيغة الوثائق المفتوحة القياسية (التي تستخدمها الحزمة منذ الإصدار الرئيسي الثاني)، والتي جعلت من الممكن الحصول على صيغة حرة كمعيار ISO لإنشاء الوثائق. لقد أجبر هذا شركة مايكروسوفت على فتح صيغتها (جزئيًّا) في الإصدارات بدءًا من Office 2007 لتتضمن صيغ OpenXML.[5]

- توفير بدائل صالحة: يجب أن يكون للبرمجية التي سنتوقف عن استخدامها بدائل تقوم بنفس وظيفة النظام السابق. لمعظم التطبيقات بديل واحد أو أكثر بوظائف مماثلة – إن لم تكن أفضل –. يمكنك إيجاد العديد من القوائم (شبه الكاملة) على الإنترنت لتطبيقات متوفرة لجنو/لينكس تكافئ وظائف تطبيقات وندوز.

- دعم التطبيقات العاملة لأنظمة أخرى: في بعض الحالات، يمكن تشغيل تطبيقات لأنظمة يونكس

---

5 لمشروع المكتب الحر LibreOffice من The Document Foundation المشتق من مشروع المكتب المفتوح مساهمات قوية بهذا الصدد، حيث جرت تحسينات كبيرة جداً على دعم صيغ مكتب مايكروسوفت، إضافة إلى التحسينات الكبيرة في الأداء. أما بالنسبة لمكتب مايكروسوفت، فقد صار يدعم الصيغة الحرة ODF في إصدار 2010 وما يليه.

أخرى (بنفس المعمارية، مثل Intel x86)، أو لنظام MS-DOS أو وندوز، عبر حزم توافقية أو نوع من المحاكيات.

معظم المشاكل التي تؤثر على الهجرة في الأنظمة المكتبية يتم حلها بشكل بطيء لكنه ثابت، وسيسمح لنا مستقبلاً بالحصول على عدد كبير من المستخدمين المكتبيين لجنو/لينكس، والذين سيصلون إلى تطبيقات أفضل مع ازديادهم سيشجعون شركات البرمجيات على بدء تنفيذ إصدارات لجنو/لينكس.

في حالة الشركات، يمكن عمل هذا بهجرة سلسة بدءًا بالخوادم ومحطات العمل، تلي ذلك الحواسيب المكتبية بعد اتباع برنامج تدريب مكثف للمستخدمين في الأنظمة والتطبيقات الجديدة.

ومن العمليات التي ستساعد بشكل كبير تقديم البرمجيات مفتوحة المصدر في التعليم وفي الإدارة العامة، كما في حالة إقليم إكستريمادورا في إسبانيا وتوزيعة جنو/لينكس المسماة Linex؛ أو الإجراءات الحديثة لأخذ هذه البرمجيات إلى التعليم الأساسيّ، أو الإجراءات المتبعة من الجامعات بتطبيق مساقات أو مواضيع تستخدم هذه الأنظمة.

5 ورشة عمل في الهجرة: تحليل دراسة حالة

سنحاول في ورشة العمل هذه تطبيق ما تعلمناه في هذه الوحدة لتحليل بعض عمليات الهجرة البسيطة، وبعض التفاصيل عن التقنيات المطلوبة (في حالة تقنيات الشبكة، سنبحث فيها في الوحدة المتعلقة بإدارة الشبكة).

سنأخذ بعين الاعتبار دراسات الحالات التالية:

1) الهجرات الفردية للمستخدمين المكتبيين لوندوز إلى نظام جنو/لينكس.

2) هجرات المؤسسات الصغيرة بأنظمة وندوز وقليل من يونكس.

3) هجرة خادم مستقل يعمل بوندوز إلى خادم سامبا يعمل على جنو/لينكس.

5.1 الهجرات الفردية للمستخدمين المكتبيين لوندوز إلى نظام جنو/لينكس.

يفكر المستخدم بالهجرة إلى جنو/لينكس. عادة، يكون هناك مدة من التواجد المشترك في البداية، ولهذا يكون لدى المستخدم كلا النظامين، ويستخدم كلاً منهما لعدد من المهام: المهام سيستمر عملها على وندوز، بينما يتعلم المستخدم النظام الجديد، ويجد برمجيات مكافئة أو برمجيات جديدة تقوم بمهام لم يكن هناك برمجية متاحة للقيام بها في السابق.

الهجرة بالنسبة لمستخدم شخصي على الأغلب واحدة من أعقد العمليات؛ نحتاج لتقديم بدائل للمستخدمين لما يستخدمونه في العادة، ليكون التكيف معه بأسهل ما يمكن، بحيث يمكن للمستخدم الانتقال بشكل تدريجي وبسهولة إلى النظام الجديد.

الاحتمال الأول سيكون تثبيتاً مزدوجاً للنظام الأصلي (وندوز) مع نظام جنو/لينكس.

الخطوة الأولى لتحديد إعدادات الجهاز ستتكون من التأكد بأنه متوافق مع لينكس، إما من قائمة بالعتاد المتوافق، أو بالتأكد من المصنع إذا ما كان هناك مكونات جديدة نحتاج لشرائها أو إذا كانت المكونات الحالية تحتاج إعداداً محدّدًا. إذا لم نكن نعرف عتادنا، فيمكننا تفقده من "إدارة الأجهزة" في وندوز (في لوحة التحكم)، أو باستخدام نوع من برمجيات التعرف على العتاد. وفي نفس الوقت، هناك طريقة ينصح باتباعها، وهي استخدم قرص حي لتوزيعة جنو/لينكس، والتي ستسمح لنا

بتفحص عمل جنو/لينكس على عتادنا دون الحاجة لتثبيته بالفعل، حيث المتطلب الوحيد هو إمكانية إقلاع النظام من قرص CD/DVD (في بعض الحالات سنحتاج لتغيير إعدادات BIOS لكي تعمل). هناك أقراص حية مع دعم ممتاز لفحص العتاد، ومعظم توزيعات جنو/لينكس توفر قرصاً حياً لتفحص عمله مبدئيًّا (في بعض الحالات كأبونتو، يمكن عمل التثبيت الكامل باستخدام نفس القرص الحي). على أيّة حال، علينا أن نذكر أن الفحص باستخدام قرص حي معين لا يعني أنه لن تكون هناك مشكلات مع التثبيت النهائي، وذلك لأنه [أحيانا] لا يكون القرص الحي لنفس توزيعة جنو/لينكس التي سنثبتها في النهاية، أو لأن إصدارات النظام و/أو التطبيقات لن تكون نفسها.

فيما يخص التثبيت الحقيقي على القرص، سنحتاج إلى مساحة حرة غير مجزأة من القرص، أو إذا كان لدينا أقسام من نوع FAT32 فيمكننا تحرير مساحة باستخدام برمج تجعل من الممكن ضبط حجم الأقسام، وتصغير القرص الموجود (من البديهي أن يكون عمل نسخة احتياطية هنا من الأمور التي ينصح بها). الآن، تدعم معظم التوزيعات عدة تقسيمات للأقراص وأساليب تصغير الأقسام، لكن قد تظهر مشاكل اعتماداً على التوزيعة. إذا لم يكن هناك مساحة كافية، أو كان هناك أقسام بأنظمة ملفات تسبب مشاكل (مثل NTFS مع بعض التوزيعات)، فربما علينا التفكير التفكير بشراء قرص صلب إضافي لاستخدامه بأكمله أو استخدام جزء منه لجنو/لينكس.

بعد فحص العتاد، سيكون علينا تحديد توزيعة من نظام جنو/لينكس والتي سنستخدمها (هناك إمكانية ذكرناها سابقاً وهي اختيار قرص حي كان مرضياً لنا، وتثبيت تلك التوزيعة). إذا لم يكن لدى المستخدم خبرة في جنو/لينكس، أو كانت لديه معرفة محدودة بالحاسوب، فيفضل استخدام إحدى التوزيعات الأكثر قرباً للمستخدم النهائي، مثل فيدورا، أو ماندريفا، أو سوزي، أو أبونتو، أو ما شابه.[6] إذا كنا أكثر معرفة أو تستهوينا التجربة، فيمكننا تجربة توزيعة دبيان. في حالة التوزيعات التجارية، في معظم الاحيان تكون التوزيعات مع العتاد المتوافق (إصدارات الأعمال مثل ردهات وسوزي تصدر شهادات للعتاد الذي تدعمه) تثبت بشكل مثالي ودون أية مشكلة، والإعدادات الأساسية مضبوطة لتسمح لنظام التشغيل أن يستخدم مباشرة. أثناء العملية، سيكون علينا تثبيت البرمجيات، والتي عادة ما تكون محددة ضمن مجموعات من البرمجيات الموجهة:

---

6 عادة ما أنصح المستخدمين العرب باستخدام توزيعات لينكس ذات الطابع العربي والإسلامي، وذلك لأنها أقرب لبيئتنا وتفهم احتياجاتنا، وعادة تأتي مزودة بكل ما نحتاجه. شخصياً أنصح بأعجوبة المبنية على فيدورا، وسبيلي وهلال المبنيتين على أبونتو.

للخوادم، تطبيقات معينة أو تطبيقات مكتبية، مثل حزم المكتب، تطبيقات التطوير (إذا كنا مهتمين بالبرمجة)، وغيرها.

ما إن يتم تثبيت النظام، سيكون علينا البحث في موضوع مشاركة البيانات، كيف سنشارك البيانات بين النظامين؟ أو هل من الممكن مشاركة تطبيقات معينة؟ هناك حلول عديدة لهذا:

4) الطريقة غير المباشرة: يتم ذلك بمشاركة الملفات باستخدام قرص مرن مثلاً. لعمل هذا أفضل شيء أدوات معروفة بـ mtools، والتي تسمح بالوصول المباشر إلى الأقراص المرنة بهيئة MS-DOS، وهناك العديد من الأوامر التي تعمل بشكل مشابه جداً لما في MS-DOS ووندوز. لهذه الأوامر نفس الأسماء تماماً كما في أوامر MS-DOS الأصلية، إلا أنها مبدوأة بحرف "m"، فمثلاً: mcd, mcopy, mdir, mdel, mformat, mtype، وغيرها.

5) الطريقة المباشرة: تتم هذه باستخدام نظام ملفات وندوز مباشرة. كما سنرى في وحدة الإدارة المحلية، يمكن لجنو/لينكس قراءة وكتابة عدد كبير من أنظمة الملفات، ومن ضمنها FAT و FAT32 و NTFS (في بعض الحالات قراءة فقط، لكن معظم التوزيعات تضمّن مشغل ntfs-3g مسبقاً الذي يسمح بالكتابة). ضم قرص وندوز مطلوب في البداية، وهذا يجعل من الممكن تضمين نظام ملفات وندوز كنقطة في نظام ملفات لينكس؛ فمثلاً، يمكننا ضم قرص وندوز إلى /mnt/Windows/، ومن تلك النقطة نصل إلى مجلداته وملفاته للقراءة والكتابة. مع ملفات آسكي ASCII النصية، نحتاج لأخذ المحادثة بينهما بعين الاعتبار، حيث يتعامل يونكس ووندوز معهما بطريقة مختلفة: في يونكس، لنهاية السطر محرف واحد فقط، وهو المحرف العاشر في آسكي (تلقيم سطر)، أما في ندوز فلها محرفان، العاشر والثالث عشر (تلقيم سطر والعودة) (وكملاحظة تدعو للفضول، يستخدم ماك المحرف 13). هذا يعني أننا عادة عندما نقرأ ملفات آسكي دوس ووندوز، فهي تحوي على محارف غريبة في نهايات السطور. هناك محررات تتعامل معها مباشرة مثل emacs، وهناك على أيّة حال أدوات جنو/لينكس التي تجعل من الممكن تحويلها إلى هيئات أخرى (أدوات مثل duconv, recode, dos2UNIX, UNIX2dos).

6) استخدام تطبيقات: هناك بدائل قليلة لتشغيل تطبيقات MS-DOS ووندوز (بعضها وليس كلها). لجنو/لينكس، هناك محاكيات MS-DOS مثل Dosemu و DosBox، ولوندوز هناك برمجية واين. يمكنها تشغيل العديد من

تطبيقات وندوز (فمثلاً، يمكنها تشغيل بعض إصدارات حزمة المكتب ومتصفح Internet Explorer)، وهي تحسن باستمرار. إذا كان تشغيل تطبيقات وندوز ضرورياً، يمكن لبعض البرمجيات المملوكة مساعدتنا، تعطي هذه التطبيقات دعماً إضافياً لواين، ومن الأمثلة عليها Win4Lin و CrossOver وفي بعض الحالات دعم خاص للألعاب مثل Cedega. ومن الحلول الأخرى الممكنة استخدام جهاز تخيلي، من الأمثلة على البرمجيات المستخدمة بكثرة منها VMWare و VirtualBox، والتي تنشئ حاسوباً شخصيًّا كاملاً كجهاز وهمي، يقوم البرنامج بمحاكاته، ويمكن تثبيت عدد كبير من أنظمة التشغيل المختلفة عليه. VMWare و VirtualBox متاحان في إصدارات لوندوز وجنو/لينكس، والتي تجعل من الممكن الحصول على جنو/لينكس مثبتاً ووندوز يعمل عليه افتراضياً، أو وندوز مثبت وجنو/لينكس افتراضي. هناك أيضاً حلول أخرى لأجهزة تخيلية أخرى، مثل كيمو Qemu و KVM و Bochs. وفي أجزاء أخرى، هناك أجهزة تخيلية، أو بشكل عام محاكاة مستخدمة وموجهة لإنشاء خوادم تخيلية، مع حلول مثل خادم VMWare، أو المشاريع المفتوحة مثل Xen و OpenVZ و Vserver، والتي من الممكن فيها إنشاء العديد من الأجهزة التخيلية تعمل على نظام تشغيل موجود بشكل مشترَك (عادة عبر تعديلات على النواة التي تدعم هذه المحاكاة)، أو حتى مباشرة على العتاد، مع طبقات صغيرة من البرمجيات.

إضافة إلى مشاركة المعلومات (التطبيقات و/أو البيانات)، يمكنك البحث عن تطبيقات جنو/لينكس تستبدل تطبيقات وندوز الأصلية حيث يتعلم المستخدم استخدامها تدريجياً، ويرى أنها توفر الوظائف المطلوبة.

مثال

من الحالات الاعتيادية حزم المكتب التي يمكن ترحيلها إلى المكتب المفتوح، والذي له توافقية كبيرة مع ملفات Office ويعمل بشكل مشابه له، أو Koffice (لواجهة كدي)، أو Gnumeric و AbiWord (لجنوم). أو – في حالة معالجة الصور – يمكننا أخذ جمب، ذي المزايا المشابهة لفوتوشوب. والعديد من مشغلات الوسائط، مثل Xine و Mplayer. على الإنترنت، يمكننا إيجاد العديد من القوائم لبرامج متكافئة لكل من وندوز وجنو/لينكس.

5.2 هجرة مؤسسة صغيرة بأنظمة وندوز وقليل من يونكس

الهجرة في مؤسسة (حتى وإن كانت صغيرة) لها صعوبات عديدة: سيكون لدينا بيئات عمل مختلفة، وبرمجيات مختلطة، و – مرة أخرى – مستخدمون يقاومون التغيير.

لنفكر الآن في مؤسسة مع أجهزة وندوز وبعض أجهزة يونكس كخوادم ومحطات عمل وبعض المستخدمين "الفوضويين" نوعاً ما. فلندرس الموقف التالي على سبيل المثال: لدى المؤسسة شبكة محلية صغيرة من أجهزة وندوز، مشتركة بين المستخدمين كأجهزة متساوية في مجموعة عمل وندوز (ليس هناك نطاقات خوادم وندوز).

المجموعة متنوعة: لدينا أجهزة بوندوز 98, ME, NT, XP ولكن بإعدادات لكل مستخدم مع البرمجيات التي يحتاجها لأعماله اليومية: سواء كانت برمجيات المكتب، أو المتصفح، أو قارئ البريد الإلكتروني، أو بيئات التطوير لمختلف لغات البرمجة (مثل سي، و سي++، وجافا).

هناك بعض الموارد العتادية الإضافية المتوفرة، مثل العديد من الطابعات المرتبطة بالشبكة (تقبل أعمال TCP/IP)، والتي يمكن استخدامها من أيّ نقطة في داخل المؤسسة. في نفس الوقت، هناك جهاز مشترك، مع موارد خاصة، مثل الماسح الضوئي، وناسخ CD، ومجلدات مشتركة عبر الشبكة، والتي يمكن للمستخدمين أن يتركوا مجلدات لهم فيها، مع ملفاتهم لعمليات النسخ الاحتياطي أو لاسترجاع الصور المدخلة بالماسح الضوئي على سبيل المثال.

لدينا أيضاً العديد من محطات العمل، في هذه الحالة Sun Microsystem's SPARC والتي نستخدم Solaris عليها (وهو نسخة تجارية من يونكس من شركة Sun)، هذه المحطات مخصصة للتطوير وبعض التطبيقات العلمية والرسومية. لهذه الأجهزة خدمات NFS لمشاركة الملفات و NIS+ للتعامل مع معلومات المتسخدمين الذين يتصلون بها، والذين يمكنهم عمل ذلك من أيّ جهاز مباشرة. تتضمن بعض الأجهزة خدمات محددة؛ أحدها خادم وِب التابع للشركة، والآخر مستخدَم كخادم بريد.

نحن نبحث في إمكانية الهجرة إلى جنو/لينكس بسبب اهتمامنا بتطوير التطبيقات واهتمام خاص من بعض المستخدمين باستخدام هذا النظام.

وأيضاً، الهجرة ستتم في الغالب من أجل حل مشاكل معينة متعلقة بالأمن - بعض أنظمة وندوز القديمة ليست الحلّ الأفضل لمشاركة الملفات؛ نريد تقييد الوصول إلى الطابعة (تكلفة الورق والتكاليف المتعلقة بها مرتفعة) لحصص معقولة أكثر. وفي نفس الوقت، نرغب بأن يكون لدينا قدر معين من الحرية، لا نريد إجبارهم على تغيير النظام، لكننا سنقترح عليهم ذلك.

وأيضاً سنستغل الفرصة لشراء عتاد جديد لتدعيم العتاد الموجود، فمثلاً، إذا كانت محطات العمل تحتاج لمساحة أقراص إضافية، والتي تسبب تحديداً في البريد الإلكتروني وحسابات المستخدمين.

بعد هذا الوصف القصير لمؤسساتنا (في الحالات الأخرى الاكثر تعقيداً، قد يملأ هذا الوصف صفحات عديدة، أو قد يكون وثيقة كاملة تحلل الوضع الحالي وتنشئ مقترحاً للمستقبل)، يمكننا أن نبدأ بالتفكير في الإمكانيات لحل كلّ هذا:

7) ماذا نفعل بمحطات العمل الحالية؟ تكلفة الصيانة والتراخيص مرتفعة. نحتاج لتغطية صيانة للأعطال حال حدوثها، العتاد المكلف (في هذه الحالة، أقراص SCSI) إضافة إلى توسعات الذاكرة المكلفة. تكلفة نظام التشغيل وتحديثاته مكلفة أيضاً. في هذه الحالة، لدينا إمكانيتان (اعتماداً على الميزانية التي لدينا لإجراء التعديلات):

- يمكننا تقليل التكاليف بتحويل هذه الأجهزة إلى أنظمة جنو/لينكس. للأنظمة معمارية SPARC، وهناك توزيعات تدعم هذه المعمارية. يمكننا تبديل الخدمات إلى مكافئاتها في جنو/لينكس؛ الاستبدال يفترض أن يكون مباشراً، حيث أننا نستخدم نظام يونكس بالأساس.

- وهناك احتمالية أخرى، وهي استبعاد عتاد Sun المملوك، وتحويل المحطات إلى أجهزة شخصيّة قوية تعمل بجنو/لينكس؛ سيجعل هذا الصيانة سهلة لاحقاً، لكن تكلفة البداية ستكون مرتفعة.

8) وماذا عن برمجيات محطات العمل؟ إذا كان قد تم تطوير التطبيقات داخلياً، قد يكون تصريفها مرة أخرى، أو إجراء تعديلات بسيطة للبيئة الجديدة كافياً. إذا كانت مملوكة، فسيكون علينا أن نرى إذا كان بإمكان الشركة تقديمها لبيئات جنو/لينكس، أو إذا كان بإمكاننا إيجاد بدائل بوظائف مشابهة. في حالة المطورين، بيئاتهم للغات سي، و سي++، وجافا محمولة بسهولة؛ في حالة سي و سي++، فيمكن استخدام مصرّف سي من جنو gcc، وهناك العديد من بيئات التطوير المتكاملة IDEs للتطوير (مثل Kdevelop و Anjuta وغيرها)؛ أو في حالة جافا، فيمكن استخدام أدوات Sun على جنو/لينكس وفي بيئات عديدة مفتوحة المصدر (IBM's Eclipse أو NetBeans).

9) وماذا عن المستخدمين؟ لنسبة للمهتمين بأنظمة جنو/لينكس، فيمكننا تثبيت أجهزة مزدوجة بوندوز

وجنو/لينكس، مما يمكنهم من البدء بتجربة النظام، وإذا كانوا مهتمين، فيمكننا في النهاية الانتقال إلى نظام جنو/لينكس واحد. يمكننا أن نجد نوعين من المستخدمين: مستخدمي حزم المكتب فقط، والذين سيحتاجون للحزمة ومتصفح وبريد، وكلها يمكن توفيرها مع أسطح مكتب جنو/لينكس، مثل جنوم وكدي، وبرمجيات مثل المكتب المفتوح، ومتصفح موزيلا فيرفُكس، وبريد موزيلا المعروف بطائر الرعد Thunderbird (أو أي بديل مثل Kmail أو Evolution أو غيرها). هذه بدائل مكافئة تقريباً، وكل ذلك يعتمد على رغبة المستخدم بتجربة واستخدام البرمجيات الجديدة. بالنسبة للمطورين، يمكن أن يكون التغيير مباشراً أكثر، حيث يتوفر لهم الكثير من البيئات والأدوات المرنة؛ يمكنهم الانتقال كلياً إلى أنظمة جنو/لينكس أو العمل مباشرة مع محطات العمل.

10) والطابعات؟ يمكننا استخدام محطة عمل كخادم طباعة (سواء عبر طوابير TCP/IP أو عبر خادم سامبا)، والتحكم بالطباعة عبر الحصص.

11) والأجهزة المشتركة؟ يمكن ترك العتاد المشترك على نفس الجهاز أو التحكم به من نظام جنو/لينكس. اعتماداً على مساحة القرص المشتركة، يمكن نقلها إلى خادم سامبا يمكنه إبدال الحالي.

12) هل نزيد مساحة الأقراص؟ سيعتمد هذا على ميزانيتنا. يمكننا زيادة التحكم باستخدام نظام حصص يوزع المساحة بعدل ويفرض قيوداً على ملئها.

5.3 هجرة خادم وندوز وحيد إلى خادم سامبا يعمل على جنو/لينكس

العملية الأساسية المطلوبة أطول بكثير، راجع المراجع لمعرفة الخطوات الكاملة التي يجب اتباعها.

في هذه الحالة، العملية الأساسية المطلوبة هي الهجرة من خادم وندوز يشارك الملفات والطباعة إلى خادم سامبا على نظام جنو/لينكس.

بفضل برمجيات مثل سامبا، الهجرة من بيئات وندوز مرنة جدًّا وسريعة، بل وتحسن من أداء الجهاز.

لنفترض أن جهازاً يتبع لمجموعة عمل اسمها GROUP يشارك طابعة اسمها PRINTER وملف مشترك اسمه DATA،

والذي ما هو إلا قرص D للجهاز. يصل العديد من عملاء وندوز إلى المجلد للقراءة والكتابة، ضمن شبكة محلية بعناوين IP من 192.168.1.1 – 192.168.1.254، الأول منها لخادم وندوز هذا، وسيكون للعملاء أرقام أخرى (عادة ما تستخدم الشبكات ذات 192,168,x,x كعناوين لتركيب شبكات داخلية خاصة).

كجزء من عمليتنا هذه، سنبني خادم سامبا، والذي كما رأينا سيسمح لنا بتشغيل ميفاق SMB/CIFS في جنو/لينكس. يسمح هذا الميفاق لنظام الملفات والطابعات بالتفاعل عبر الشبكة على أنظمة تشغيل مختلفة. يمكننا ضم مجلدات تنتمي لوندوز على أجهزة جنو/لينكس، أو جزء من مجلدات جنو/لينكس على وندوز، والطابعات كذلك الأمر. يتكون الخادم من مراقبين اثنين (عمليتي نظام) تسميان smbd و nmbd.

تدير خدمة smbd طلبات العملاء من الملفات والطابعات المشتركة. بينما تدير خدمة nmbd نظام أسماء الأجهزة والموارد تحت ميفاق NetBIOS (الذي أنشأته IBM). هذا الميفاق مستقل عن الشبكة المستخدمة (في وندوز NT/2000/XP، تستخدم مايكروسوفت بشكل عام NetBios على TCP/IP). يقدم nmbd أيضاً خدمات WINS، وهي خدمة إعطاء الأسماء التي تعمل عادة على وندوز NT/Server إذا كان لدينا مجموعة من الأجهزة؛ إنها نوع من تجميع DNS و DHCP لبيئات وندوز. العملية معقدة نوعًا ما، لكن باختصار: عندما يبدأ جهاز وندوز بالعمل، أو يكون لديه عنوان IP ثابت، أو متغير عبر خادم DHCP، إضافة إلى احتمال وجود اسم NetBios (أعطاه المستخدم للجهاز: في تعريف الشبكة)، فإن عميل WINS يتصل بالخادم ليبلغ عن عنوانه؛ إذا طلب جهاز على الشبكة اسم NetBios، فسيتم الاتصال بخادم WINS للحصول على عنوان IP له، ومن ثم يتم إنشاء اتصال معه. يشغل nmbd هذه العملية على جنو/لينكس.

كأيّ خدمة شبكة أخرى، يجب أن لا نشغلها مالم نأخذ بعين الاعتبار المخاطرة التي قد تترتب على تفعيلها، وكيفية تقليل هذه المخاطرة قدر الإمكان. فيما يخص سامبا، علينا أن نكون واعين للقضايا الأمنية، لأننا نفتح جزءًا من شبكتنا المحلية أو ملفات وطابعات الشبكة. سيكون علينا أيضاً تفقد قيود الاتصالات بشكل جيد وذلك لتجنب الوصول للمستخدمين أو الأجهزة غير المرغوبة. في هذا المثال البسيط، لن نعلق على هذه القضايا؛ في الحالات الحقيقيّة، سيكون علينا دراسة الخيارات الأمنية وأن نتيح الوصول لمن نريد لهم ذلك فقط.

في عملية الهجرة، في البداية سنضبط نظام جنو/لينكس لدعم سامبا، سنحتاج لدعم أنظمة ملفات سامبا في النواة (smbfs)، والتي تكون عادة مفعلة مسبقاً. علينا أن نضيف أيضاً أن هناك دعماً إضافياً في النواة الآن عبر وحدة cifs، والتي كما في إصدار النواة 2.6.20 تعتبر الوسيلة المبدئية، تاركة smbfs كخيار ثانوي. تقدم وحدة cifs دعماً لمزايا إضافية تتعلق بميفاق CIFS (كامتداد لـ SMB). عبر أسماء أنظمة الملفات smbfs و cifs، تسمح لنا هذه الوحدات بتنفيذ عمليات لضم أنظمة ملفات وندوز إلى شجرة ملفات لينكس (mount -t smbfs أو mount -t cifs). بعيداً عن حقيقة كون دعم النواة يزداد توجهاً نحو cifs، إلا ان هناك خصائص معينة قد تتطلب دعم smbfs، مما يعني أن كلتي الوحدتين مفعلتان عادة في النواة. علينا أن نذكر أيضاً قضية الإعدادات، حيث تعتمد smbfs في عملها على إعدادات سامبا (كما سنرى في ملف smb.conf)، بينما تعطى وحدة cifs إعداداتها عبر العمليات (مثل عملية الضمّ عبر mount).

في حالة استخدام خادم سامبا، إضافة إلى دعم النواة، سنحتاج لتثبيت حزم البرمجيات ذات العلاقة: علينا أن نتأكد من الحزم المتعلقة بسامبا التي توفرها التوزيعة، وتثبيت هذه الحزم المتعلقة بعمل الخادم. وأيضاً، الحزم المتعلقة بسامبا كعميل إذا أردنا، في حال كنا نرغب بأن نكون عملاء لأجهزة وندوز أو لتجربة الموارد المشتركة مع أجهزة وندوز من نظام جنو/لينكس لدينا. في توزيعة دبيان، هذه الحزم هي: samba, samba-common, smbclient, smbfs. قد يكون من المثير للاهتمام أيضاً تثبيت swat، وهو أداة رسومية مبنية للوِب لإدارة خدمات سامبا. لخادم جنو/لينكس سامبا للمثال المطروح سيكون علينا ترحيل المحتويات من قرص D القديم (الذي كنا نحتفظ فيه بملفاتنا المشتركة) من الجهاز الأصلي إلى الجهاز الجديد، ووضع محتواه في مسار، مثل /home/DATA/، سواء عبر عمل نسخ احتياطي، أو عبر نقله عبر FTP، أو استخدام سامبا كعميل لنقل الملفات.

فيما يتعلق باستخدام جنو/لينكس كعميل سامبا، الأمر سهل نسبيًّا. باستخدام أوامر العميل للاستخدام المتقطع لنظام الملفات:

1) نضم مجلداً مشتركاً لوندوز (على اعتبار أن host اسم خادم وندوز)، في نقطة ضم موجودة محددة مسبقاً:

```
smbmount //host.carpeta /mnt/windows
```

2) سنجعل الوصول إلى مجلد وندوز للجهاز الخادم في مجلدنا المحلي، لنصل إليه في شجرة الملفات:

```
/mnt/windows/
```

3) لاحقاً، عندما ننتهي من استخدامه، يمكننا فصل المورد بتنفيذ:

```
smbumount /mnt/windows
```

إذا لم نكن نعلم المورد المشترك، فيمكننا الحصول على قائمة بتنفيذ:

```
smbclient -L host
```

ويمكننا أيضاً استخدام smbclient //host/folder وهو برنامج مماثل لعميل ftp.

وإذا أردنا جعل أنظمة الملفات متاحة دائماً، أو توفير إعدادات معينة خاصة، يمكننا دراسة استخدام mount مباشرة (أدوات smbxxxx تستخدمه)، سواء مع أنظمة ملفات smbfs أو cifs (المدعومة في النواة)، آخذين المعامِلات بعين الاعتبار (استيثاق المستخدمين/المجموعات أو معاملات الخدمة الاخرى) التي سيكون علينا توفيرها اعتماداً على الحالة، ومن إعدادات سامبا الموجودة مسبقاً.

في حالة خادم سامبا، بمجرد انتهائنا من تثبيت كل برمجيات سامبا، سيكون علينا ضبط الخادم عبر ملفات الضبط الخاصة به. اعتماداً على الإصدار (أو التوزيعة)، سيكون هذا الملف في etc/smb.conf/ أو في etc/samba/smb.conf/. الإعدادات المذكورة هنا تعود للإصدار الرئيسي الثالث من سامبا المثبت على نظام توزيعة دبيان. قد يكون للإصدارات الأخرى تعديلات طفيفة قليلة.

أثناء تثبيت حزم البرمجيات سنُسأل عادة عن بيانات تتعلق بضبطها. في حالة سامبا، سنُسأل عن مجموعة العمل التي سيخدمها، سيكون علينا وضع نفس اسم المجموعة كما في وندوز. سنُسأل أيضاً عما إذا كنا نريد كلمات سرّ مشفرة (يُنصح بها لأسباب أمنية، في وندوز 9x كان يتم إرسالها في نصّ غير مشفر، في حالة واضحة للأمن الضعيف وهشاشة كبيرة في النظام).

سنرى فيما يلي عملية ضبط الملف smb.conf. لهذا الملف ثلاثة اجزاء رئيسية:

1) عام General (خصائص العمل العامة).

2) متصفح Browser (يتحكم بما تراه الأجهزة الأخرى من مواردنا).

3) مشاركة Share (يتحكم بما نشاركه)

في الدليل المطوّل لهذا الملف، يمكننا أن نرى الخيارات المتاحة (man smb.conf). سنحرر الملف بمحرر ونرى بعض سطور الملف (المحارف '#' و ';' في بداية الملف تعليقات: إذا كان السطر يحوي ';' في بدايته فهو تعليق؛ لتفعيل سطر، إذا كان سطر إعداد اختياري، علينا تحريره وإزاله ';' من بدايته):

```
workgroup = GROUP
```

يرينا هذا مجموعة عمل وندوز التي سيكون أجهزة عملاء وندوز أعضاء فيها.

```
Server string = %h server (Samba %v)
```

يمكننا وضع وصف نصيّ لخادمنا. حرفا h و v الذان يظهران هما متغيرا سامبا الذان يوفران اسم المضيف وإصدار سامبا. لأسباب أمنية، من الجيد عدم استخدام خيار v، حيث سيُعلِم هذا من في الخارج بإصدار سامبا الذي لدينا؛ إذا كانت هناك علل أمنية معروفة، يمكن استخدامها.

```
Hosts allow = 192.168.1
```

قد يكون أو لا يكون هذا السطر موجوداً، ويمكننا إضافته لتفعيل خيار المضيفين الذين سيُخدَمون؛ في هذه الحالة، كل العناوين من 192.168.1.1 إلى 192.168.1.255.

```
printcap name = /etc/printcap
```

ملف printcap هو مكان احتفاظ جنو/لينكس بتعريف الطابعات، وهو المكان الذي سيبحث فيه سامبا عن معلومات عنها.

```
guest account = nobody
```

هذا حساب الضيف. يمكننا إنشاء حساب مختلف، أو يكفي تفعيل الوصول إلى سامبا للمستخدمين المسجلين في نظام جنو/لينكس.

```
Log file = /var/log/samba/log.%m
```

سيخبرنا هذا السطر بمكان احتفاظ سامبا بالسجلات. يسجل واحد لكل عميل (المتغير m هو اسم العميل المتصل).

```
Encrypt password = true
```

لدواعي أمنية، يُنصح باستخدام كلمات سرّ معمّاة. إذا كان لدينا أجهزة عميلة بنظام وندوز 98 أو NT والأحدث. كلمات السر هذه مخزنة في الملف /etc/samba/smbpasswd، والمُنشأ عادة لمستخدمي سامبا المثبّت. يمكن تغيير كلمات السر بالأمر smbpasswd. هناك أيضاً خيار اسمه UNIX password sync، الذي يسمح للتغيير بأن يتم لكلتي كلمتي السرّ (مستخدم سامبا ومستخدم لينكس) معاً.

بعد ذلك، سنقفز إلى قسم تعاريف المشاركة Share definitions:

```
[homes]
```

تسمح هذه السطور بالوصول إلى حسابات المستخدمين من أجهزة وندوز. إذا لم نكن نريد هذا، فسنضيف ';' إلى بدايات هذه السطور، وعندما تتصل الأجهزة سترى تعليق الاسم. الكتابة معطلة مبدئيًّا، لتشغيلها ما عليك سوى ضبط خيار الكتابة إلى "yes".

عند أي مشاركة لمجلد محدد (يطلق سامبا لفظ قسم partition على مجموعة البيانات المشتركة)، سنستمر كما نرى في الأمثلة الظاهرة (فمثلاً انظر إلى تعريف مشاركة القرص المضغوط في السطور التي تبدأ بـ [cdrom]). في المسار سنضع مسار الوصول.

مثال

في حالتنا – على سبيل المثال – يمكننا إعطاء الاسم DATA للقسم على المسار /home/DATA/، والذي نسخنا فيه القرص D من جهاز وندوز الأصلي، والمسار الذي يمكن إيجاده فيه، إضافة إلى مجموعة كبيرة من الخيارات التي يمكن تعديلها، كالمستخدمين المصرح لهم الوصول إليها، وطريقة عمل ذلك.

هناك أيضاً تعريف ملفات شخصية، مما يجعل من الممكن التحكم بالملفات الشخصة لمستخدمي وندوز، وبعبارة أخرى،

المجلد المحفوظ فيه إعداد سطح مكتب وندوز، وقائمة بدء التشغيل، وغيرها.

الأمر مشابه بالنسبة للطابعات: يُنشأ قسم باسم الطابعة (نفس الاسم المُعطى في جنو/لينكس)، وفي المسار نضع عنوان الطابور المرتبط بالطابعة (سنجده في جنو/لينكس في /var/spool/samba/PRINTER). والخيار printable = yes إذا أردنا أن تُرسَل الوظائف عبر سامبا. ويمكنا أيضاً تقييد وصول المستخدمين (المستخدمون المخولون).

ما إن نتِم هذه التغييرات فسيكون علينا فقط حفظها وإعادة بدء سامبا ليتمكن من قراءة الإعداد الجديد. في دبيان:

```
/etc/init.d/samba restart
```

الآن سيكون مجلدنا وطابعتنا المشتركان عبر سامبا متاحين لخدمة المستخدمين دون أن يلاحظوا أي فرق مقارنة باتصالاتهم السابقة مع نظام خادم وندوز.

أنشطة

1) في وصف خدمات جنو/لينكس، هل نجد أننا نفتقد وظيفة معينة؟ ما أنواع الخدمات الأخرى التي يمكن أن نضيفها؟

2) في دراسة الحالة الثانية في الشرح (المتعلقة بالمؤسسة)، كيف ستغير البنية التحتية لتقنية المعلومات إذا كانت ميزانيتك صفراً؟ وإذا كانت ميزانيتك عادية؟ وإذا كانت ميزانيتك كبيرة؟ اعرض بعض الحلول البديلة لما ذُكِر.

3) تقنيات المحاكاة، مثل VMWare Workstation و VirtualBox – الأجهزة التخيلية عبر برمجيات – التي يمكنها تثبيت أنظمة تشغيل على حاسوب شخصي تخيلي. يمكنك الحصول على هذه البرمجيات من www.vmware.com أو www.virtualbox.org. اختبر (إذا كان لديك ترخيصاً لوندوز) تثبيتها على وندوز، ومن ثم على جنو/لينكس على الجهاز التخيلي (أو بالعكس). ما الفوائد التي تقدمها هذه الطريقة لمشاركة أنظمة التشغيل؟ ما المشاكل التي تسببها؟

4) إذا كان لدينا جهازان لتثبيت خادم سامبا، يمكننا اختبار تثبيت وإعداد الخادم في إعدادات عميل يونكس سامبا وخادم وندوز، أو عميل وندوز وخادم سامبا على جنو/لينكس. يمكنك اختبارها على جهاز واحد باستخدام الجهاز نفسه كخادم وعميل سامبا.

المراجع

مصادر أخرى للمراجع والمعلومات

[LDP] يقدم مشروع توثيق لينكس دروساً تتعلق بنواحٍ مختلفة لنظام جنو/لينكس، ومجموعات من الأدلة الأكثر تفصيلاً.

[Mor03] مرجع جيد لإعدادات أنظمة لينكس، مع دراسات حالات في بيئات مختلفة؛ يعلق على توزيعات مختلفة من دبيان وردهات.

أدوات أساسيّة للمدير

د. جُسِـــپ جُبرا إستيڤَ

مقدمة

على مدير أنظمة جنو/لينكس أن يقوم يومياً بعدد كبير من المهام. بشكل عام، لا تقوم فلسفة لينكس على وجود أداة واحدة لكل مهمة، أو وجود طريقة واحدة للقيام بالأمور. من الشائع في لينكس توفير عدد كبير من الأدوات البسيطة بدرجات متفاوتة للقيام بالمهام المختلفة.

ستكون هناك تركيبة من الأدوات الأساسية، ولكل واحدة منها مهمة محددة وواضحة ستسمح لنا بحل المشكلات أو القيام بالمهام الإدارية.

سنرى في هذه الوحدة مجموعات مختلفة من الأدوات، ونحدد بعض مهامها الأساسية، وسنرى بعض الأمثلة على استخداماتها. في البداية، سنختبر بعض معايير عالم جنو/لينكس، والتي ستساعدنا في إيجاد بعض الخصائص الأساسية التي نتوقعها في أيّ توزيعة جنو/لينكس. تخبرنا هذه المعايير مثل LSB (أو قاعدة معيار لينكس Linux Standard Base) و FHS (معيار هيكلية نظام الملفات Filesystem Hierarchy Standard) عن الأدوات التي يمكننا توقع إيجادها متاحة، وعن بنية شائعة لنظام الملفات، والعادات العديدة التي تحتاج التوزيعة لاتباعها لتعتَبر نظام جنو/لينكس، للاحتفاظ بقوانين مشتركة للتوافقية بينها وبين غيرها.

لأتمتة المهام الإدارية نستخدم عادة أوامر مجموعة في نصوص شِل (shell scripts، وتعرف أيضاً بنصوص الأوامر)، بلغة تترجمها صدفة النظام (مترجم الأوامر). يسمح لنا في البرمجة بلغات الصدفة هذه بإضافة أوامر النظام إلى بنية تحكم انسيابية، وبهذا يكون لدينا بيئة للنماذج الأولية السريعة من الأدوات لأتمتة المهام.

وهناك مخطط آخر شائع، وهو استخدام أدوات لتصريف وتصحيح اللغات عالية المستوى (مثل سي). عامةً، سيستخدمها مدير النظام لإنشاء تطويرات جديدة للتطبيقات أو الأدوات، أو لدمج التطبيقات التي تأتي كمصدر برمجي والتي تحتاج لأن يتم تطويعها وتصريفها.

سنحلّل أيضاً استخدام بعض الأدوات الرسومية آخذين بعين الاعتبار سطور الأوامر الاعتيادية. هذه الادوات تسهل المهام الإدارية، لكن استخدامها محدود نظراً لاعتمادها الكبير على توزيعات جنو/لينكس وإصداراتها. رغم هذا، هناك

أدوات مفيدة يمكن تصديرها بين التوزيعات.

وفي النهاية، سنحلل مجموعة من الادوات الضرورية لإبقاء النظام محدثاً، ألا وهي أدوات إدارة الحزم. البرمجيات التي تقدمها توزيعة جنو/لينكس أو المضافة لاحقاً. عادة ما تقدم في وحدات تسمى "حُزَم"، والتي تتضمن ملفات برمجية معينة، إضافة إلى الخطوات العديدة اللازمة لتحضير التثبيت ومن ثم إعداده (في بعض الاحيان) أوتحديث أو إزالة برمجية معينة. وتحمل كل توزيعة برمجية إدارية للاحتفاظ بقوائم للحزم المثبتة أو التي يمكن تثبيتها، إضافة إلى التحكم بالإصدارات الموجودة أو الاحتمالات العديدة لتحديثها عبر المصادر الأصلية المختلفة.

1 الأدوات الرسومية وسطر الأوامر

هناك عدد كبير من الأدوات – والتي سنختبر جزءًا صغيراً منها في هذه الوحدة والوحدات التالية لها – والتي تتوفر كأدوات إدارية من أطراف خارجية، مستقلة عن التوزيعة، أو من موزّع جنو/لينكس نفسه.

وتتفاوت هذه الأدوات في مقدار النواحي التي تغطيها من مهمة إدارية معينة، ويمكن أن تظهر مع واجهات عديدة مختلفة: سواء كأدوات سطر الأوامر مع العديد من خيارات الإعداد و/أو الملفات أو الأدوات النصية مع نوع من القوائم؛ أو كأدوات رسومية مع واجهات مناسبة أكثر للتعامل مع المعلومات، ومعالجات لأتمتة المهام، أو واجهة إدارة وِب.

يوفر لنا كلّ هذا مدى واسعاً من الاحتمالات التي تهتم بها الإدارة، لكن سيكون علينا دائماً تقييم مدى سهولة استخدامها والاستفادة من استخدامها، ومعرفة المدير المسؤول عن هذه المهام.

يمكن أن تتضمن مهام مدير نظام جنو/لينكس الشائعة العمل مع توزيعات مختلفة (مثل التي سنناقشها كفيدورا أو دبيان أو أي توزيعة أخرى) أو حتى العمل مع أشكال أخرى من أنظمة يونكس المملوكة. سيتبع هذا وجوب اتخاذ طريقة معينة للعمل تسمح لنا بالقيام بالمهام في الأنظمة المختلفة بطريقة موحدة.

لهذا السبب، سنحاول خلال الوحدات المختلفة توضيح أكثر النواحي شيوعاً والمهام الإدارية التي ستستخدم أكثر في مستوى أدنى، عبر سطر الأوامر و/أو تحرير ملفات إعداد مرتبطة.

تحوي أي توزيعة لينكس سطر أوامر أو أدوات نصية أو رسومية لإكمال ما ذكر أعلاه وتسهيل هذه المهام الإدارية بدرجات متفاوتة. لكن علينا أخذ عدد من الأشياء بعين الاعتبار:

1. هذه الأدوات واجهات متقنة نوعاً ما لأدوات سطر الأوامر الأساسية وملفات الإعداد المتعلقة بها.
2. لا توفر هذه الأدوات عادة كل المزايا أو الإعدادات التي يمكن الحصول عليها من المستوى المنخفض.
3. يمكن أن لا يكون قد تمّ التعامل مع الأخطاء جيداً، أو يمكن أن تتوفر ببساطة رسالة من نوع "تعذر تنفيذ هذه المهمة".

4. يخفي استخدام هذه الادوات العمل الداخلي – أحياناً بشكل كامل – للمهمة أو العملية. وجود فهم جيد للعمل الداخلي أمر أساسي بالنسبة للمدير، خاصة إذا كان المدير مسؤولاً عن تصحيح الأخطاء أو تحسين الخدمات.
5. هذه الادوات مفيدة لتحسين الإنتاجية عندما تتوفر المعرفة اللازمة عند المدير للتعامل مع المهام الروتينية بفعالية أعلى ولأتمتتها.
6. أو على العكس، يمكن أن تكون المهام معقدة جدًّا، وتتطلب الكثير من المعامِلات لإنتاج كثير من البيانات، مما قد يجعل من المستحيل التحكم بها يدوياً. في هذه الحالة، يمكن أن تكون الأدوات عالية المستوى مفيدة جداً وتجعل من الممكن تنفيذ المهام التي يصعب التحكم بها دونها. على سبيل المثال، قد يتضمن هذا التصنيف أدوات مرئية، وأدوات مراقبة، واختصاراً للمهام أو الخدمات المعقدة.
7. لأتمتة المهام، هذه الأدوات (بمستوى أعلى) قد لا تكون مناسبة: قد لا تكون مصممة للخطوات التي تحتاج لتنفيذها، أو قد تقوم بها دون كفاءة. فمثلاً، قد تكون من الحالات إنشاء مستخدمين، حيث يمكن أن تكون الاداة الرسومية جذابة بسبب طريقة إدخال البيانات، لكن ماذا لو كنا نريد إدخال قائمة بعشرات أو مئات المستخدمين بدل واحد أو مجموعة قليلة منهم؟ إذا لم تكن معدة لذلك، فستكون دون كفاءة إطلاقاً!
8. وفي النهاية، يرغب المدراء دائماً بأن يخصصوا مهامهم باستخدام أدوات يرونها أكثر كفاءة وسهولة للتكيّف معها. في هذه الناحية، من الشائع استخدام أدوات أساسية ذات مستوى منخفض، ونصوص شِل (سندرس الأساسيات في هذه الوحدة) ودمجها لتشكيل مهمة.

يمكننا استخدام هذه الأدوات أحياناً (أو يومياً) إذا كان لدينا المعرفة اللازمة وذلك للتعامل مع الأخطاء التي يمكن أن تنشأ أو لتسهيل عملية صممت الأداة لها، لكنها أيضاً تتحكم بالمهام التي ننفذها والمعرفة التقنية التي تندرج تحتها.

2 المعايير

المعايير، سواء كانت معايير يونكس العامة، أو كانت خاصة لجنو/لينكس، فإنها تسمح لنا باتباع بعض القواعد التي تقودنا في تعلم طريقة تنفيذ مهمة توفر لنا المعلومات الأساسية لبدء عملنا.

يمكننا في جنو/لينكس إيجاد معايير – كمعيار هيكلية نظام الملفات FHS – تخبرنا بما يمكننا إيجاده في هيكلية نظام ملفات نظامنا (أو أين نبحث)، أو قاعدة معيار لينكس LSB التي تناقش المكونات المختلفة التي نجدها في الأنظمة.

يصف معيار هيكلية نظام الملفات FHS بنية شجرة نظام الملفات الرئيسية "/"، التي تحدد بنية المجلدات والملفات الرئيسية التي ستحويها. يستخدم هذا المعيار أيضاً - نوعاً ما - في قطاع من أنظمة يونكس التجارية، حيث كانت هناك اختلافات كثيرة جعلت كل مصنع يغير البنية كما يريد. المعيار الذي تم تصوره أساساً لجنو/لينكس تم عمله لإصلاح هذا الوضع وتجنب التغييرات الكبيرة. رغم هذا، يتّبع هذا المعيار بدرجات متفاوتة، فتتبع معظم التوزيعات نسبة كبيرة من هذا المعيار، حيث يقومون بتغييرات طفيفة، أو يضيفون ملفات أو مجلدات غير موجودة في المعيار.

يمكن أن يكون مخطط المجلدات الأساسي كالتالي:

1. /bin/: أدوات النظام الأساسية، وعادة ما تكون برامج يستخدمها المستخدمون، سواء من أوامر النظام الأساسية (مثل bin/ls/ التي تعرض محتوى مجلد) أو الصدفات (مثل bin/bash/) أو غيرها.
2. /boot/: المجلدات الضرورية لإقلاع النظام، مثل صورة نواة لينكس في boot/vmlinuz/.
3. /dev/: سنجد هنا ملفات خاصة تمثل الأجهزة المختلفة في النظام، الوصول إلى الطرفيات في أنظمة يونكس يتم كما لو كانت ملفات. يمكننا إيجاد ملفات مثل dev/console/، و dev/modem/ و dev/mouse/ و dev/cdrom/ و dev/floppy/ ... والتي يمكن أن تكون روابط لملفات محددة أكثر للأجهزة ذات نوع مشغل أو الواجهة يستخدمها الجهاز: dev/mouse/ المربوط بـ dev/psaux/، والذي يمثل فأرة من نوع PS2؛ أو dev/cdrom/ المربوط بـ dev/hdc/، وهو مشغل اقراص مضغوطة موصول بمتحكم IDE الثاني لجهاز رئيسي.

نجد هنا أجهزة IDE مثل dev/hdx/ و سكزي مثل dev/sdx/ ... بحيث يتغير x حسب رقم الجهاز. علينا أن نذكر هنا أن هذا المجلد كان في البداية ثابتاً، والملفات كانت معرفة مسبقاً، و/أو مضبوطة في أوقات معينة، لكن الآن نستخدم أساليب لتقنية مرنة (مثل hotplug أو udev) يمكنها اكتشاف الأجهزة وإنشاء ملفات /dev/ ديناميكياً عندما يقلع النظام أو وهو يعمل، عند إضافة أو إزالة إجهزة.

4. /etc/: ملفات الإعداد. ستحتاج معظم المهام الإدارية لاختبار وتعديل الملفات المضمنة في هذا المجلد، فعلى سبيل المثال: etc/passwd/ يحوي جزءاً من معلومات حسابات المستخدمين في النظام.
5. /home/ (المنزل): يحوي حسابات المستخدمين، ونعني هنا المجلدات الشخصية لكل مستخدم.
6. /lib/: مكتبات النظام التي تشترك بها برامج المستخدمين، سواء كانت ثابتة (امتداد a.) أو متغيرة (امتداد so.). على سبيل المثال، مكتبة سي القياسية في ملفات libc.so أو libc.a. وعلى التحديد، يمكننا عادة إيجاد الوحدات المتغيرة (الديناميكية) لنواة لينكس في lib/modules/.
7. /mnt/: نقطة لضم أنظمة الملفات مؤقتاً باستخدام الأمر mount، مثلاً: /mnt/cdrom/ لضم قرص ضوئي موجود في قارئ الأقراص المضغوطة مؤقتاً.
8. /media/: نقطة ضم شائعة للأجهزة الممكن إزالتها.
9. /opt/: البرمجيات المضافة إلى النظام بعد التثبيت عادة تأتي هنا [وهي غالباً برمجيات من أطراف خارجية لا تأتي من مدير الحزم]؛ وهناك مكان آخر صالح للتثبيتات وهو /usr/local/.
10. /sbin/: أدوات النظام الأساسية. وهي أدوات محجوزة للمدير (المستخدم الجذر). فمثلاً: sbin/fsck/ المستخدم لتصحيح حالة نظام الملفات.
11. /tmp/: الملفات المؤقتة للتطبيقات أو النظم نفسه. رغم أنها للعمل مؤقتاً بين التنفيذين لبرنامج أو خدمة لا يمكننا أن نفترض بأنها ستجد الملفات السابقة.
12. /usr/: عناصر مختلفة مثبتة في النظام. بعض برمجيات النظام المكتملة أكثر مثبتة هنا، إضافة إلى كماليات الوسائط المتعددة (الأيقونات، والصور، والأصوات، مثلاً في /usr/share/) وتوثيق النظام (/usr/share/doc/). تستخدم أيضاً في /usr/local/ لتثبيت البرمجيات [غالباً للتثبيت اليدوي وإضافة النصوص

البرمجية].

13. /var/: ملفات من نوع تقرير أو حالة و/أو ملفات الخطأ للنظام نفسه وللخدمات العديدة للنظام والشبكة. على سبيل المثال، ملفات التقارير في /var/log/، ومحتوى البريد في /var/spool/mail/، ووظائف الطباعة في /var/spool/lpd/.

هذه بعض المجلدات المعرفة في FHS لنظام الملفات، ثم تحدد بعض التصنيفات الفرعية مثلاً، مثل محتوى /usr/ و /var/ ، والبيانات الاعتيادية و/أو الملفات التنفيذية يتوقع أن تكون أقل ما يمكن في هذه المجلدات (انظر إلى مراجع توثيق FHS).

فيما يخص التوزيعات، تتبع فيدورا/ردهات معيار FHS بشكل قريب جدًّا. تعرض فقط تغييرات بسيطة في الملفات المقدمة في /usr/ و /var/. وفي /etc/ مجلد للمكون الذي يمكن ضبطه، وليس في /opt/ و /usr/local/ برمجيات مثبتة إلا إذا ثبتها المستخدم. دبيان تتبع المعيار، لكنها تضيف بعض مجلدات الضبط الخاصة في /etc/.

والمعيار الثاني الذي سنذكره هو LSB (قاعدة معيار لينكس). مهمته تحديد مستويات التوافقية بين التطبيقات، والمكتبات، والأدوات، لتكون محمولية التطبيقات بين التوزيعات المختلفة ممكنة دون مشاكل كثيرة. إضافة إلى هذا المعيار، توفر مجموعات اختبارات لفحص مستوى التوافقية. LSB في ذاته مجموعة من المعايير المختلفة المطبقة على جنو/لينكس.

3 توثيق النظام

من النواحي الهامة في مهامنا الإدارية الحصول على التوثيق الصحيح لنظامنا والبرمجيات المثبتة. هناك عدد من مصادر المعلومات، لكن علينا إبراز بعضها.

1. **man** هو - إلى حدٍّ بعيد - الخيار الأفضل للمساعدة. يسمح لنا بالعودة إلى دليل جنو/لينكس المجمع في أقسام عديدة تتعلق بالأوامر الإدارية، وهيئات الملفات، وأوامر المستخدمين، استدعاءات لغة سي، وغيرها. عادة، للحصول على المساعدة ذات العلاقة بأمر، فسنحصل على ما يكفي باستخدام:

   man command

2. تصف كل صفحة عادة الأمر مع خياراته و - عادة - عدداً من الأمثلة على الاستخدام. في بعض الأحيان قد يكون هناك أكثر من مدخلة في الدليل. فمثلاً، قد يكون هناك استدعاء سي بنفس اسم أمر؛ في هذه الحالة سيكون علينا تحديد أي قسم نريد النظر فيه:

   man n command

   حيث يكون n رقم القسم.

   هناك أيضاً العديد من الأدوات لاستكشاف الأدلة، مثل xman و tkman التي تساعد على تفحص الأجزاء المختلفة وفهارس الأوامر بواجهة رسومية. وهناك أمر آخر ملفت للانتباه وهو كلمة apropos والتي تساعدنا على تحديد مكان صفحات man التي تتحدث عن موضوع معين (له علاقة بالكلمة).

3. Info هو نظام مساعدة شائع آخر. طوِّر هذا البرنامج في جنو لتوثيق كثير من أدواته. هو بالأساس أداة نصية يمكن البحث فيها عن أجزاء وصفحات باستخدام نظام تنقل بسيط يعتمد على لوحة المفاتيح.

4. توثيق التطبيقات: إضافة إلى صفحات man معينة، من الشائع تضمين توثيق إضافي في التطبيقات على هيئة دروس، أو شروحات، أو أدلة مستخدمين. عادة ما تكون مكونات التوثيق هذه مثبتة في المجلد /

   usr/share/doc/ (أو /usr/doc/) حسب التوزيعة، حيث عادة يُنشأ مجلد لكل حزمة تطبيق (يمكن أن يكون للتطبيق في العادة حزمة توثيق منفصلة).

5. الأنظمة الخاصة بالتوزيعات. تأتي توزيعة ردهات بعدة أقراص مضغوطة للأدلة المرجعية التي يمكن تثبيتها في

النظام والتي تأتي بهيئة HTML أو PDF. لفيدورا مشروع توثيق على موقعها. توفر دبيان أدلتها بهئية حزمة برمجيات إضافية تُثبَّت عادة في /usr/doc/. في نفس الوقت، فيها أدوات لتصنيف التوثيق في النظام وتنظيمه باستخدام القوائم والمرئيات، مثل dwww أو dhelp التي توفر واجهات وِب لاختبار توثيق النظام.

6. وأخيراً، أسطح مكتب X – مثل جنوم وكدي – تحوي عادة نظام توثيقها وأدلتها الخاصة، إضافة إلى المعلومات من المطورين، سواء كملفات مساعدة مرئية في تطبيقاتها التي تجمع كل ملفات المساعدة (مثل devhelp في جنوم).

4 البرمجة بلغة الصدفة

يستخدم المصطلح العام "صدفة" أو شِل shell للإشارة إلى برنامج يخدم كواجهة بين المستخدم ونواة نظام جنو/لينكس. في هذا القسم، سنركز على الصدفات النصية التفاعلية التي سنجدها كمستخدمين عند ولوجنا إلى النظام.

الصدفة أداة نظام تسمح للمستخدمين بالتفاعل مع النواة عبر ترجمة الأوامر التي يدخلها المستخدم في سطر الأوامر أو الملفات التي من نوع نص شِل برمجي.

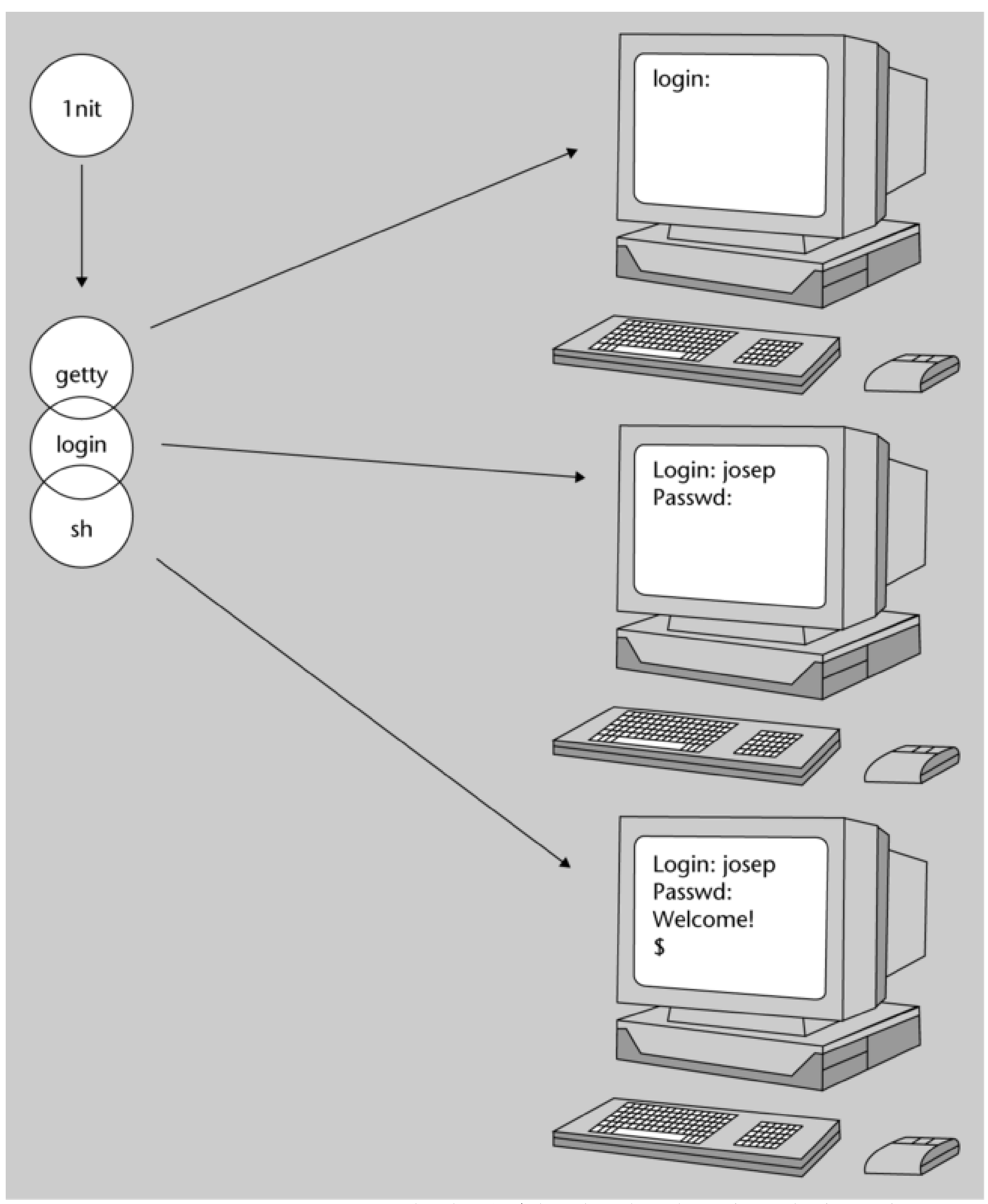
الصدفة هي ما يراه المستخدمون من النظام، بينما تبقى بقية نظام التشغيل مخفية عنهم. تُكتب الصدفة بنفس طريقة كتابة عملية مستخدم (برنامج)؛ لا تشكل جزءًا من النواة، بل تعمل كبرنامج آخر للمستخدم فقط.

عندما يقلع نظام جنو/لينكس لدينا، فإنه يوفر للمستخدمين واجهة بمظهر محدد؛ قد تكون الواجهة نصية أو رسومية. اعتماداً على أوضاع أو مستويات الإقلاع، سواء مع أوضاع الواجهة النصية أو تلك التي تعطينا إقلاعاً مباشراً واجهة رسومية عبر X.

في أوضاع الإقلاع إلى واجهة رسومية، تتكون الواجهة من مدير وصول لإدارة إجراء ولوج المستخدمين باستخدام صفحة غلاف رسومية تسأل عن المعلومات اللازمة للدخول: معرف المستخدم وكلمة مروره. مدراء الوصول في جنو/لينكس عدّة: xdm (الذي تعود ملكيته لـ X Window)، و gdm (لجنوم)، و kdm (لكدي)، إضافة إلى بضعة مدراء أُخَر مرتبطين بمدراء نوافذ مختلفين. بمجرد ولوجنا، سنجد أنفسنا في واجهة X Window الرسومية مع مدير نوافذ [أو بيئة سطح مكتب] مثل جنوم أو كدي. للتفاعل مع صدفة تفاعلية، كل ما سنحتاج لعمله فتح واحدة من برامج محاكاة الطرفيات المتوفرة.

إذا كان وصولنا في الوضع النصي، وبمجرد ولوجنا، فسنحصل على وصول إلى صدفة تفاعلية.

وهناك حالة أخرى للحصول على صدفة تفاعلية، وهي بالوصول إلى الجهاز عن بعد، سواء عبر أيّ من الإمكانيات النصّية مثل telnet, rlogin, ssh أو عبر الإمكانيات الرسومية مثل محاكي X Window.



شكل 1: مثال على تشغيل صدفة نصية وعمليات النظام ذات العلاقة

4.1 الصدفات التفاعلية

عند بدء الصدفات التفاعلية يرى المستخدم محثًّا يدل على إمكانية إدخال سطر أوامر. تصبح الصدفة بعد دخولها مسؤولة عن تفعيل وتشغيل العمليات المطلوبة في عدد من المراحل:

1. قراءة وترجمة سطر الأوامر.
2. تقدير المحارف الخاصة wildcards مثل * ? $ وغيرها.
3. إدارة تحويلات الدَخْل/الخرْج المطلوبة، والأنابيب "|" والعمليات الخلفية (&).
4. التعامل مع الإشارات.
5. التحضير لعمل البرامج.

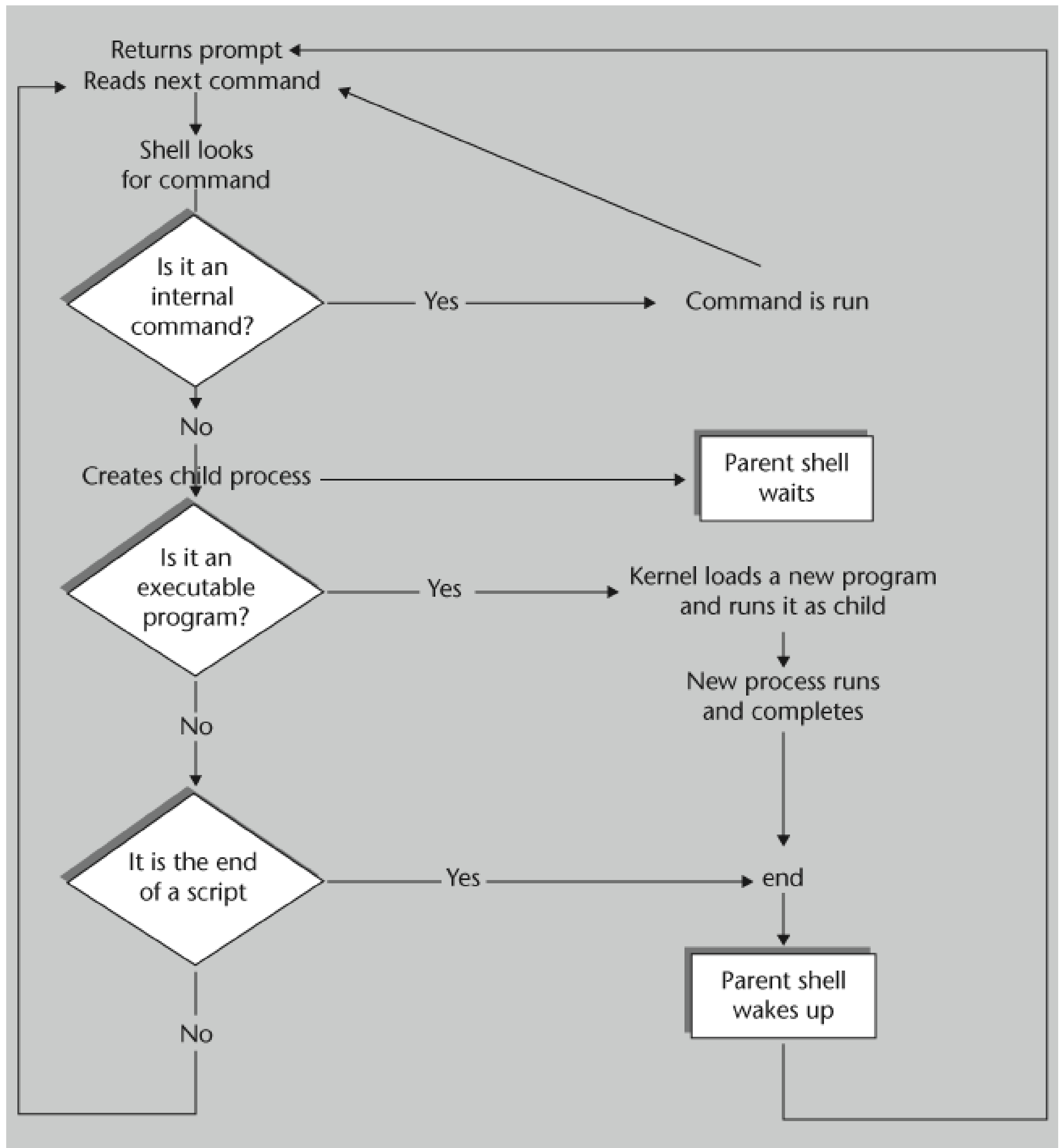
تكون سطور الأوامر عادة طرقاً لتشغيل أوامر النظام، أو أوامر الصدفة التفاعلية، أو تشغيل التطبيقات، أو نصوص الصدفة البرمجية.

> نصوص الصدفة البرمجية ملفاتٌ نصية تحوي تسلسل أوامر للنظام، إضافة إلى سلسلة من الأوامر الداخلية للصدفة التفاعلية، إضافة إلى بنية التحكم اللازمة لمعالجة سير البرنامج (من نوع while و for وغيرها).

يستطيع النظام تشغيل ملفات النصوص البرمجية مباشرة باستخدام اسم الملف. لتشغيلها، نستحضر الصدفة مع اسم الملف أو نعطي النص البرمجي بلغة الصدفة صلاحيات التنفيذ.

إلى حد ما، سنرى نصوص شِل ككود للغة برمجة مفسّرة تُنفَّذ على الصدفة التفاعلية ذات العلاقة. نصوص الصدفة مهمة جداً للمدير، وبشكل أساسي لسببين:

1. إعدادات النظام ومعظم الخدمات تتوفر عبر أدوات على هيئة نصوص شِل.
2. الطريقة الرئيسية لأتمتة العمليات الإدارية هي إنشاء نصوص شِل.



شكل 2: مخطط تحكم شِل بسيط

كل الملفات التي تستدعيها عملية الصدفة لها ثلاثة ملفات معرفة مسبقاً، تحددها متحكمات handles الملفات المرتبطة.

مبدئيًّا، هذه هي الملفات:

1. الدَخل القياسي: عادة تكون موكلة إلى لوحة مفاتيح الطرفية (أو الواجهة النصية)؛ تستخدم متحكم الملف رقم 0 (استخدم في ملفات يونكس متحكمات ملف من أعداد صحيحة).

2. الخَرج القياسي:موكلة عادة إلى شاشة الطرفية، تستخدمتحكم الملف 1.

3. الخطأ القياسي: عادة ما يكون موكلاً إلى شاشة الطرفية، يستخدم متحكم الملف 2.

يخبرنا هذا بأن أي برنامج يعمل من الطرفية سيكون لديه مبدئياً ملف إدخال مرتبط بلوحة مفاتيح الطرفية، وملف خَرج مرتبط بالشاشة، وأنه سيرسل الأخطاء إلى الشاشة.

وأيضاً، توفر الصدفات الآليات الثلاثة التالية:

1. **إعادة التوجيه**: بافتراض أن التعامل مع أجهزة الدَخل/الخَرج والملفات يتم بنفس الطريقة في يونكس، فالطرفية – ببساطة – تتعامل معها كملفات. من وجهة نظر المستخدم، يمكن إيكال متحكمات الملفات بحيث تسير البيانات من متحكم ملف إلى أي متحكم ملف آخر؛ يطلق على هذا إعادة التوجيه. على سبيل المثال، نتعامل مع إعادة توجيه متحكمات الملفات 0 أو 1 كإعادة توجيه الدَخل/الخَرج القياسي.
2. **الانابيب**: يمكن استخدام الخرج القياسي لبرنامج كدخل برنامج آخر باستخدام الأنابيب. يمكن ربط العديد من البرامج ببعضها باستخدام الأنابيب لإنشاء ما يعرف بخط الأنابيب pipeline.
3. **تزامن** برامج المستخدم: يمكن للمستخدمين تشغيل عدّة برامج في نفس الوقت، مشيرين إلى أنها ستنفّذ في الخلفية، أو في المقدمة، مع تحكم حصري بالشاشة. وتتكون طريقة أخرى من السماح للعمليات الطويلة بالعمل في الخلفية، بينما نتفاعل مع الصدفة والبرامج الأخرى في المقدمة.

عمليًّا، تتضمن هذه الصدفات في جنو/لينكس:

1. **إعادة التوجيه**: يمكن لبرنامج استقبال دخل وخرج من ملفات أو أجهزة أخرى.

مثال لنرَ:

command op file

حيث يمكن أن يكون op واحداً مما يلي (الرأس المبب يحدد الوجهة):

- < : يستقبل إدخالاً من ملف.
- > : يرسل خرجاً إلى ملف.
- >> : تعني إضافة الخرج (حيث الإشارة المفردة > تعيد إشاء الملف).

2. الأنابيب: بعمل سلسلة من أوامر عديدة ونقل بياناتها:

```
command1 | command2 | command3
```

3. تخبرنا هذه التعليمات أن الأمر الأول ربما سيستقل الأوامر من لوحة المفاتيح، وإرسال مخرجاته إلى الامر الثاني، الذي سيستقبلها ويرسل مخرجاته إلى الأمر الثالث، والذي بدوره سيستقبلها ويرسلها إلى الخرج القياسي (مبدئيًّا: الشاشة).
4. التزامن في الخلفية: سيعمل أي أمر يتم تنفيذه بوجود & في آخره في الخلفية، والواجهة النصية للصدفة ستعود فوراً، بينما يستمر تنفيذ البرنامج. يمكننا تتبع عمل الأوامر عبر أمر ps وخياراته، الذي ستسمح لنا بمراقبة حالة عمليات النظام. ولدينا أيضاً أمر القتل، الذي يسمح لنا بإنهاء العمليات التي ما تزال تعمل أو التي دخلت حالة خطأ: يسمح لنا kill -p PID بقتل عملية، حيث يمثل PID رقم تعريف العملية. PID هو المعرّف المرتبط بالعملية، وهو عدد صحيح أوكله النظام إليها، ويمكن الحصول عليه باستخدام الأمر ps.

4.2 الصدفات

يسمح لنا استقلال الصدفات عن نواة نظام التشغيل (الصدفة مجرد طبقة واجهة) الحصول على عدد منها على النظام. لكن من أكثر هذه الصدفات استخداماً ما يلي:

1. باش Bash (اختصار لـ Bourne-Again Shell). صدفة جنو/لينكس المبدئية.
2. صدفة بورن Bourne Shell (وهي sh). كانت هذه دائماً صدفة يونكس القياسية، والتي تضمنتها كلّ أنظمة يونكس في بعض الإصدارات. عادة تكون صدفة المدير (الجذر) المبدئيّة. في جنو/لينكس تحل باش Bash مكانها، وهي إصدار محسّن من صدفة بورن التي انشأها ستيفن بورن في AT&T في نهاية السبعينات. المحثّ المبدئيّ فيها هو '$' (للمستخدم الجذر هي #).
3. صدفة كورن Korn shell (وهي ksh). هي مجموعة من صدفة بورن (تم الاحتفاظ فيها ببعض التوافقية) – كتبها ديفيد كورن في AT&T (في منتصف الثمانينيات) – والتي تحتفظ ببعض مزايا صدفتي بورن و C، مع بعض الإضافات. المحث المبدئيّ هو $.
4. صدفة سي (وهي csh). طورها بِل جوي في جامعة بيركلي قرابة نهاية السبعينات، وبها إضافات ملفتة مقارنة ببورن – مثل سجل الأوامر، والأسماء المستعارة (أو الاختصارات alias)، والحساب من سطر الأوامر – وتكمل أسماء الملفات وتتحكم بالوظائف في الخلفية. المحثّ المبدئي للمستخدمين هو '%'. يفضل مستخدمو يونكس هذه الصدفة للتفاعل مع النظام، لأن النصوص البرمجية فيها مضغوطة أكثر وتنفّذ أسرع. وفي ذات الوقت، من مزايا النصوص البرمجية لصدفة سي – كما يشير اسمها – هو أن السياق فيها مبني على لغة سي (رغم أنها ليست نفسها).
5. صدفات أخرى، كالإصدارات المقيدة أو المتخصصة من المذكورة أعلاه.

لقد تزايدت أهمية باش Bourne Again Shell – Bash منذ تضمينها في أنظمة جنو/لينكس كصدفة مبدئية. تشكل هذه الصدفة جزءًا من مشروع برمجيات جنو. إنها محاولة لدمج الصدفات الثلاثة السابقة (صدفة بورن و سي و كورن)، مع الاحتفاظ بسياق صدفة بورن الأصلية. هذه هي الصدفة التي سنركز عليها في أمثلتنا اللاحقة.[1]

1 هناك أيضاً صدفة أخرى تُدعى zsh قد ترغب عزيزي القارئ بتجربتها والاطلاع على خصائصها.

هناك طريقة سريعة لمعرفة الصدفة التي نتواجد فيها كمستخدمين باستخدام المتغير SHELL$ باستخدام تعليمة سطر الاوامر:

echo $SHELL

سنجد أن بعض النواحي مشتركة في كلّ الصدفات:

1. كلها تسمح بكتابة نصوص تنفيذية (سكربتات شِل)، والتي يتم تفسيرها وتنفيذها باستخدام الاسم (إذا كان للملف صلاحيات تنفيذ) أو بتمريره كمعامل لأمر الصدفة.
2. لمستخدمي النظام صدفة مبدئية مرتبطة بهم. تقدّم المعلومات عند إنشاء حساب المستخدم. سيوكل المدير صدفة لكل مستخدم، أو – إذا لم يفعل – ستوكل الصدفة المبدئية (مثل باش في جنو/لينكس). تُحفَظ تلك المعلومات في ملف كلمات المرور etc/passwd/، ويمكن تغييرها بالأمر chsh. نفس هذا الأمر مع خيار l- سيعرض صدفات النظام المتاحة (انظر أيضاً إلى etc/shells/).
3. كل صدفاً هي فعليًّا ملف تنفيذيّ، وعادة تكون موجودة في مجلدات /bin/ في جنو/لينكس (أو في /usr/bin/).
4. يمكن كتابة نصوص الصدفة في أيّ منها، لكن بما يتناسب مع قواعد سياق كلّ منها، والذي عادة ما يكون مختلفاً (وأحياناً تكون الاختلافات بسيطة). سياق الإنشاء والأوامر الداخلية موثّقون في صفحات أدلة الاستخدام man لكلّ صدفة (مثلاً man bash)
5. لكل صدفة ملفات بدء مرتبطة بها (تسمى ملفات البدء)، ويمكن لكل مستخدم ضبطها بناء على احتياجاته، بما يتضمن الأكواد، والمتغيرات، والمسارات ...
6. تكمن إمكانيات البرمجة في تركيب سياق كلّ صدفة (من إنشائها)، وال Hوامر الداخلية لكلّ صدفة، ومجموعة من أوامر يونكس شائعة الاستخدام في النصوص البرمجية مثل: cut, sort, cat, more, echo, grep, wc, awk, sed, mv, ls, cp, ...
7. إذا كنّا نستخدم صدفة معينة، فلا شيء يمنعنا من بدء نسخة جديدة من الصدفة (نسميها صدفة فرعية) سواء كانت نفسها أو واحدة أخرى. ببساطة، نستدعيها باستخدام اسم الملف التنفيذي، سواء sh, bash, csh, ksh. وأيضاً،

عندما نشغل نصًّا برمجياً بلغة الصدفة، فإن نسخة فرعية تعمل بالصدفة المناسبة لتنفيذ النصّ المطلوب.

من الاختلافات الأساسية بين الصدفات ما يلي:

باش هي الصدفة الأساسية في جنو/لينكس (مالم يُذكر غير ذلك عند إنشاء حساب المستخدم). في أنظمة يونكس الأخرى تكون صدفة بورن (sh). باش متوافقة مع sh، وتتضمن أيضاً مزايا الصدفتين الأخريين، csh و ksh.

ملفات البدء: لصدفة sh و ksh ملف profile. (في حساب المستخدم، ويتم تنفيذه عند ولوج المستخدم)، ولصدفة كورن ksh أيضاً ملف اسمه kshrc. وهو نصّ تنفيذي، وتستخدم csh ملف login. (تعمل عند ولوج المستخدم مرة واحدة فقط)، و logout. (يُنفّذ عند الخروج من جلسة المستخدم) و cshrc. (المشابه لـ profile. في كلّ صدفة سي فرعيّة منشأة). وتستخدم باش ملف bashrc. و bash_profile. ويمكن أيضاً للمدير وضع متغيرات ومسارات شائعة في ملف /etc/profile الذي سينفَّذ قبل الملفات التي يملكها كلّ مستخدم. ملفات بدء الصدفة توضع في حساب المستخدم عند إنشائه (عادة تكون منسوخة من مجلد /etc/skel/) حيث يمكن للمدير ترك القالب العام (أو "الهيكل العظمي") للملفات المعدّة.

نصوص إعداد النظام والخدمات تكتب عادة بصدفة بورن (sh)، حيث تستخدمها معظم أنظمة يونكس بهذه الطريقة. يمكننا في جنو/لينكس أيضاً أن نجد بعضها مكتوبة بطريقة باش، وأيضاً بلغات نصية أخرى غير مرتبطة بالصدفة، مثل بيرل أو بايثون.

يمكننا تحديد الصدفة التي يعمل بها النصّ البرمجي باستخدام الأمر file، مثلاً file sciptname، أو بقراءة السطر الأول للملف، والذي يكون #!/bin/name، حيث name هو bash, sh, csh, ksh ... يخبرنا هذا السطر - عند تشغيل النصّ - بالصدفة التي يحتاجها ليتم تنفيذه (وبعبارة أخرى، أيّ صدفة فرعية سيكون علينا تشغيلها لتنفيذه؟). من المهم أن تحويه كل النصوص البرمجية، وعدا ذلك، سيحاول النظام استخدام الصدفة المبدئية (في حالتنا هذه: باش)، والسياق قد لا يكون صحيحاً، متسبباً بأخطاء كثيرة في السياق أثناء التنفيذ.

4.3 متغيرات النظام

من متغيرات النظام المفيدة (يمكننا رؤيتها باستخدام أمر echo مثلاً) التي يمكن الرجوع إليها في سطر الأوامر أو خلال برمجة نصوص الصدفة:

المتغير	مثال على قيمته	الوصف
HOME	/home/abdo/	المجلد الرئيسي للمستخدم (المنزل)
LONGNAME	Abdalraheem	هوية المستخدم عند الولوج
PATH	/usr/local/bin:/usr/bin:/bin/	المسارات (للملفات التنفيذية)
SHELL	/bin/bash/	صدفة المستخدم
PS1	$	محثّ الصدفة (يمكن للمستخدم تغييره)
MAIL	/var/mail/abdo/	مجلد البريد الإلكتروني
TERM	Xterm	نوع الصدفة المستخدمَة
PWD	/home/abdo/	المسار الحالي[2]

يمكن رؤية المتغيرات المختلفة للبيئة باستخدام الأمر env. مثلاً:

2 تم تصحيح هذه النقطة. الكتاب الأصلي يذكر بانها تعرض مسار المستخدِم الحالي، بينما الصحيح هو أنها تعرض المسار الحالي الذي نتصفحه.

```
$ env
SSH_AGENT_PID = 598
MM_CHARSET = ISO-8859-15
TERM = xterm
DESKTOP_STARTUP_ID =
SHELL = /bin/bash
WINDOWID = 20975847
LC_ALL = es_ES@euro
USER = juan
LS_COLORS = no = 00:fi = 00:di = 01;34:ln = 01;
SSH_AUTH_SOCK = /tmp/ssh-wJzVY570/agent.570
SESSION_MANAGER = local/aopcjj:/tmp/.ICE-unix/570
USERNAME = juan
PATH=/soft/jdk/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games MAIL =
/var/mail/juan
PWD = /etc/skel
JAVA_HOME = /soft/jdk
LANG = es_ES@euro
GDMSESSION = Gnome
JDK_HOME = /soft/jdk
SHLVL = 1
HOME = /home/juan
GNOME_DESKTOP_SESSION_ID = Default
LOGNAME = juan
DISPLAY = :0.0
COLORTERM = gnome-terminal
XAUTHORITY = /home/juan/.Xauthority
_ = /usr/bin/env
OLDPWD = /etc
```

4.4 البرمجة النصيّة في باش

سنلقي نظرة هنا على المفاهيم الأساسية لنصوص الصدفة في باش، وننصح بالقراءة أكثر حول الموضوع في المراجع [Bas] و [Coo].

يجب أن تبدأ كل مخطوطات باش بالسطر:

```
#!/bin/bash
```

يعبر هذا السطر عن الصدفة التي سيستخدمها المستخدم، وهي الصدفة الفعالة أثناء التنفيذ، والصدفة التي تحتاجها لتنفيذ النص الذي يلي هذا السطر.

يمكن تنفيذ النصّ البرمجي بطريقتين مختلفتين:

1. بتفيذه مباشرة من سطر الأوامر، في حال كان له صلاحيات تنفيذ. إذا لم يكن كذلك، فيمكننا إعطاؤه الصلاحيات بالأمر:

```
chmod +x script
```

2. بتنفيذه من خلال الصدفة، حيث نستدعي الصدفة بشكل مباشر:

```
/bin/bash script
```

علينا الأخذ بعين الاعتبار أنه بغض النظر عن طريقة التنفيذ، فإننا دائماً ننشئ صدفة فرعية يعمل فيها النصّ البرمجي.

4.4.1 المتغيرات في باش

يتم إسناد القيم للمتغيرات بالأمر:

```
variable = value
```

يمكن رؤية قيمة المتغير بالأمر:

```
echo $variable
```

حيث تعود بنا '$' إلى قيمة المتغير.

المتغير المبدئي مرئيّ فقط داخل النص البرمجي (أو في الصدفة). إذا كنا بحاجة لجعل المتغير مرئيا خارج السكربت، على مستوى الصدفة أو أيّ صدفة فرعية، فهذه تنشأ بطريقة استدلالية، سنحتاج لتصديره إضافة إلى إسناد قيمة له. يمكننا عمل أحد شيئين:

1. الإسناد أولاً، ثم التصدير:

```
var = value
export var
```

2. التصدير أثناء الإسناد:

```
export var = value
```

لدينا في نصوص باش البرمجية بعض المتغيرات المضبوطة مسبقاً يمكن الوصول إليها:

1. $1 – $N: تحفظ المدخلات السابقة كمعاملات للنص البرمجي من سطر الأوامر.
2. $0: تحفظ اسم النص البرمجي، وتكون المعامل 0 من سطر الأوامر.
3. $*: تحفظ كل المعاملات من 1 إلى N في هذا المتغير.
4. $: تحفظ كلي المعاملين، لكن في تنصيص مزدوج (كهذا: " ") لكل منهما.
5. $?: "status": تحتفظ بالقيمة المعادة من أحدث أمر منفّذ. مفيد لفحص حالة الخطأ، حيث يعيد يونكس 0 إذا كان التنفيذ صحيحاً، وقيمة أخرى كرقم يعبر عن الخطأ.

وهناك أمر آخر مهم بخصوص الإسناد، وهو استخدام علامات التنصيص:

6. تسمح علامة التنصيص المزدوجة بالتعامل مع كل شيء داخلها كوحدة واحدة.

7. علامات التنصيص المزدوجة تشبهها، لكنها تتجاهل المحارف الخاصة داخلها.

8. علامات التنصيص المائلة (كهذه: `command`) تستخدم لتقييم ما بداخلها إذا كان هناك تنفيذ أو استبدال ما يفترض أن يتمّ. أولاً يتم تنفيذ ما بداخلها، ثم يستبدل ما بداخل علامة التنصيص بنتيجة التنفيذ. على سبيل المثال، الأمر `var = `ls قائمة الملفات التي بداخل المجلد (نتيجة تنفيذ الأمر ls) بداخل المتغير $var.

4.4.2 مقارنات

بالنسبة للحالة، يتم استخدام test expressions أو باستخدام التعبير [expression] مباشرة.

يمكننا تجميع الحالات المتاحة كما يلي:

1. المقارنات الرقمية: -eq, -ge, -gt, -le, -lt, -ne، والتي تعني (على الترتيب): يساوي "="، أكبر من أو يساوي، أكبر من، أصغر من أو يساوي، أصغر من، لا يساوي.
2. مقارنة سلاسل المحارف: =: و =! و n- و z-، والتي تعني: يساوي، يختلف عن، ذو طول أكبر من 0، الطول يساوي صفراً أو المدخلة فارغة.
3. مقارنة الملفات: -d, -f, -r, -s, -w, -x. الملف هو: مجلد، ملف عادي، قابل للقراءة، ليس فارغاً، قابل للكتابة، قابل للتنفيذ.
4. المتغيرات الثنائية بين التعبيرات: !, -a, -o، للتعبيرات NOT (النفي) و AND (الدالة a-) و OR (الدالة o-).

4.4.3 بُنى التحكُّم

فيما يخص البرمجة الداخلية للنص البرمجي، نحتاج لأن نفكر بأننا بالأساس سنجد ما يلي:

1. أوامر من نظام التشغيل نفسه.
2. أوامر داخلية لباش (انظر إلى: man bash).
3. بُنى تحكم برمجية (مثل for و while وغيرها)، مع سياق باش.

السياق الأساسي لبُنى التحكم كالآتي:

1. البُنية: if ... then تقدّر التعبير، وإذا تم الحصول على قيمة معينة فسيتم تنفيذ الأمر.

```
If [expression]
    then
    commands
fi
```

2. البُنية if ... then ... else تُقيّم التعبير، وإذا تم الحصول على القيمة المطلوبة، فسيتم تنفيذ الأمر command1، عدا ذلك سينفَّذ الأمر command2:

```
if [expression ]
    then
        command1
    else
        command2
fi
```

3. البُنيَة if ... then ... else if ... else، وهي نفس سابقتها، مع بُنية if إضافية:

```
if [ expression ]
    then
        commands
    elif [ expression2 ]
    then
        commands
    else
    commands
fi
```

4. البُنية case select هي بنية للاختيار المتعدد بناء على قيمة الخيار

```
case string1 in
    str1)
        commands;;
    str2)
        commands;;
    *)
        commands
esac
```

5. حلقة for كبديل عن كل متغير في قائمة:

```
for var1 in list
do
    commands
done
```

6. حلقة while عند تحقق الشرط:

```
while [ expression ]
do
    commands
done
```

7. حلقة until إلى أن يتم تحقق الشرط:

```
until [ expression ]
do
    commands
done
```

8. الإعلان عن وظائف functions:

```
fname ( ) {
    commands
}
```

أو بالإعلان المتضمّن لاستدعاءات:

```
fname2 ( arg1, arg2, … argN )  {
    commands
}
```

واستدعاء الوظيفة يتم كالتالي: fname أو fname2 p1 p2 p3 … pN

5 أدوات إدارة الحُزَم

في أي توزيعة، الحُزم هي العناصر الأساسية للتعامل مع مهام تثبيت البرمجيات الجديدة، أو تحديث الموجودة، أو إزالة البرمجيات غير المستخدمة.

> بالأساس، الحزمة مجموعة من الملفات التي تشكل تطبيقاً أو مجموعة من عدد من التطبيقات ذات العلاقة، وتشكل عادة ملفاً واحداً (يعرف بمصطلح حزمة)، بهيئته الخاصة، وعادة ما يكون مضغوطاً، ويتم توزيعه عبر الأقراص الضوئيّة أو بالتنزيل من خدمة (مستودعات http و ftp).

استخدام الحزم مفيد لإضافة وإزالة البرمجيات، لأنها تُعدّ وحدة واحدة، بدلاً من الاضطرار للتعامل مع ملفات منفردة.

في محتوى التوزيعة (أقراصها الضوئية) تصنَّف الحُزَم في تصنيفات مثل: أ. حزم أساسية base: وهي حزم ضرورية لعمل النظام (أدوات، وبرامج إقلاع، ومكتبات نظام)؛ ب. نظام: أدوات إدارة، أدوات سطر أوامر؛ ج. تطوير: أدوات برمجة: محررات، مصرفات، مصححات، ...؛ د. رسوميات: متحكمات وواجهات الرسوميات، سطح المكتب، مدراء النوافذ، ...؛ هـ. تصنيفات أخرى.

عادة، لتثبيت حزمة، علينا اتباع سلسلة من الخطوات:

1. خطوات الابتداء (ما قبل التثبيت): التأكد من وجود البرمجيات المطلوبة (والإصدار المناسب) اللازمة لعملها (الاعتماديات)، سواء مكتبات النظام أو التطبيقات الأخرى التي يستخدمها البرنامج.
2. فكّ ضغط محتوى الحزمة، ونسخ الملفات إلى الأماكن المحددة لها، سواء كانت محددة بالضبط (بمكان محدد تماماً) أو يمكن نقلها إلى مجلدات أخرى.
3. ما بعد التثبيت: وضع اللمسات الأخيرة على الملفات اللازمة، وضبط المعاملات المحتملة للبرمجية، وضبطها لتناسب النظام ...

اعتماداً على أنواع الحزم، يمكن أن تكون هذه العمليات معظمها تتم آليًّا (هذه هي حالة RPM و DEB) أو قد تحتاج

لعملها كلها يدوياً (كما في حالة tgz.) وهذا يعتمد على الأدوات التي توفرها التوزيعة.

فيما يلي، سنرى ما يمكننا اعتباره أكثر الحزم التقليدية الشائعة في معظم التوزيعات. لكل توزيعة واحدة قياسية، وتدعم واحدة من الأخرى.

5.1 حزمة TGZ

ربما حزم TGZ هي أطول هذه الحزم عمراً. استخدمتها أقدم توزيعات جنو/لينكس لتثبيت البرمجيات، والعديد من التوزيعات ما زالت تستخدمها (مثل سلاكوير) وبعض أنظمة يونكس المملوكة. هي مجموعة من الملفات المجمعة بأمر tar في ملف واحد ذي امتداد tar. تم ضغطها بأداة gzip وتظهر بامتداد tgz. أو tar.gz. . في نفس الوقت، من الشائع هذه الأيام إيجاد tar.bz2. التي تستخدم أداة أخرى بدل gzip، وهي أداة bzip2، والتي توفر في بعض الأحيان ضغطاً أقوى.

على النقيض مما قد يبدو، فإن هذه الهيئة مستخدمة بكثرة، خاصة في أوساط مُنشئي وموزّعي البرمجيات خارج التوزيعة. كثير من منشئي البرمجيات الذين يعملون لمنصات عديدة، كالعديد من أنظمة يونكس المملوكة، وتوزيعات مختلفة من جنو/لينكس يفضلونها كنظام سهل وأكثر محمولية.

ومن الأمثلة على هذه الحالة مشروع جنو الذي يوزع برمجياته بهذا الشكل (على هيئة مصدر برمجي)، حيث يمكن استخدامها في أيّ يونكس، سواء نظام مملوك منه، أو أحد أشكال BSD، أو توزيعة جنو/لينكس.

إذا كان بهيئة ثنائيّة، فسيكون علينا أن نبقي في بالِنا أن يكون البرنامج مناسباً لنظامنا، فمثلاً، من الشائع استخدام طائفة من البرامج تشبه ما يلي (في هذه الحالة، الإصدار 1.4 من متصفح موزيلا [وهو سلف فيرفُكس]):

mozilla-i686-pc-linux-gnu-1.4-installer.tar.gz

حيث لدينا اسم الحزمة وهو mozilla مصممة لمعمارية i686 (فئة Pentium II والأحدث منها والمتوافق معها)، ويمكن أن تكون i386، أو i586، أو i686، أو k6 (لمعالجات amd k6)،أو k7 (لمعالجات amd athelon)، أو amd64 أو x86_64 (لمعالجات AMD64 وبعض معالجات 64bit من إنتل التي تحوي em64t)، أو ia64 (إنتل إيتانيوم Itanium)، وغيرها لمعماريات الأجهزة الأخرى، مثل: sparc, powerpc, mips, hppa, alpha, ...، ثم تخبرنا أنه للينكس، على جهاز حاسوب شخصي PC، وإصدار 1.4 للبرنامج.

إذا كانت في هيئة مصدرية، فستكون كالتالي:

mozilla-source-1.4.tar.gz

التي نرى فيها كلمة source والتي تعني "المصدر"، في هذه الحالة لا تذكر إصدار معمارية الجهاز، مما يعني أنها جاهزة ليتم تصريفها على معماريات مختلفة.

لو ذكرت معمارية الجهاز هنا، فسيعني هذا وجود أكواد مختلفة لكل نظام تشغيل أو مصدر: جنو/لينكس، Solaris, Irix, BSD ...

العملية الأساسية مع هذه الحزم تتكون من:

1) فكّ ضغط الحزمة (لا تستخدم مسارات كاملة، مما يعني أنه يمكن فكّها في أي مكان):

```
tar -zxvf file.tar.gz
```

(ويمكن استخدام نفس الأمر مع ملفات tgz.)

نستخدم مع أمر tar خيار z الذي يعني فكّ الضغط، و x الذي يعني استخراج الملفات، و v الذي يعرض العملية، و f لتحديد الملف الذي سنتعامل معه.

ويمكن أيضاً عملها دون خيار z

```
gnuzip file.tar.gz
```

(ينتج لنا ملف tar)

```
tar -xvf file.tar.gz
```

2) ما إن ننتهي من فك ضغط ملف tgz، سنحصل على الملفات التي يحويها، ويفترض عادة أن يحوي البرنامج ملفاً من نوع readme أو install، والتي تحدد خيارات التثبيت خطوة بخطوة، وكذلك الاعتماديات البرمجية إن وجدت.

يفترض بنا أولاً أن نتفقّد الاعتماديات لنرى إن كان لدينا البرمجيات الصحيحة، وإن لم يكن، ابحث عنها وثبتها.

إذا كانت حزمة ثنائيّة، فالتثبيت عادة ما يكون سهلاً نسبيًّا، حيث سيكون البرنامج إما قابلاً للتنفيذ من المكان الذي فككناه فيه (أيًّا كان هذا المكان)، أو سيحوي مثبته الخاص. وهناك احتمال آخر، وهو أن نكون مضطرين لتثبيته يدويًّا، مما يعني أنه سيكفي نسخه (بالأمر cp -r الذي ينسخ المجلد بما يحويه)، أو نقله (بالأمر mv) إلى المكان الذي نرغب بوجوده فيه.

والحالة الأخرى هي هيئة المصدر البرمجي. في هذه الحالة، سيكون علينا تصريف البرنامج قبل تثبيته. لعمل هذا، سنحتاج لقراءة التعليمات التي تكون مرفقة مع البرنامج بقدْرٍ من التفصيل. لكن معظم المطورين يستخدمون نظاماً من جنو يُدعى autoconf (وهو اختصار لكلمة autoconfiguration، أي الضبط الآليّ)، والذي يستخدم عادة الخطوات التالية (إذا لم تظهر أخطاء):

1. configure/.: نصّ برمجي يضبط الكود لنتمكن من تصريفه على جهازنا ويتأكد من وجود الادوات اللازمة. الخيار '= prefix--' (الذي يأتي متبوعاً بمسار مجلد) يجعل من الممكن تحديد مكان تثبيت البرنامج.
2. make: عملية التصريف ذاتها.
3. make install: تثبيت البرنامج في المكان الصحيح، وعادة ما يكون قد حُدِّد مسبقاً كخيار إعداد أو أُبقِيَ كما كان في الخيار المبدئي.

هذه عملية عامة، لكنها تعتمد على البرنامج إذا كان يتبعها أم لا، وهنا حالات أسوأ حيث تحتاج للقيام بالعملية بأكملها يدويًّا، بوضع لمساتك على ملفات الإعداد أو ملف makefile، و/أو تصريف الملفات كلًّا على حدة، لكن لحسن الحظ أن شيوع هذه الحالة يتضاءل مع الوقت.

في حال أردنا حذف كلّ البرمجيات المثبتة، فسيكون علينا استخدام مزيل التثبيت إذا توفر، عدا ذلك سيكون علينا حذف المجلدات والملفات المثبتة، والبحث عن الاعتماديات المحتملة.

حزم tgz شائعة نوعاً ما كآلية للنسخ الاحتياطيّ للمهام الإدارية، وذلك مثلاً لعمل نسخ عن البيانات الهامّة، وعمل نسخ احتياطية لحسابات المستخدمين، أو حفظ نسخ قديمة للبيانات لا نعلم إذا كنا سنحتاجها مرة اخرى. تُستَخدَم العملية التالية: لنفرض أننا نرغب بحفظ نسخة عن المجلد "dir"، يمكننا أن نكتب: tar -cvf dir.tar dir (الخيار c يعني أرشِف compact المجلد dir إلى الملف dir.tar)، ثم gzip dir.tar (لنضغط الملف)، أو بسطر واحد كالتالي:

```
tar -cvzf dir.tgz dir
```

ستكون النتيجة ملف dir.tgz. سنحتاج لأن نكون متنبّهين إذا اردنا الاحتفاظ بخصائص الملفات وصلاحيات المستخدمين، إضافة إلى الروابط التي قد تكون موجودة (يجب علينا أن نتأكد من خيارات الأمر tar لتتناسب مع خيارات

النسخ الاحتياطي المطلوبة).

5.2 فيدورا/ردهات: حزم RPM

يمثل نظام حزم RPM الذي أنشأته ردهات خطوة للأمام، حيث يتضمن إدارة مهام ضبط واعتماديات البرمجيات. وأيضاً، يحتفظ هذا النظام بقاعدة بيانات صغيرة بالحزم المثبتة مسبقاً، والتي يمكن الرجوع إليها وتحديثها عند كل عملية تثبيت جديدة.

من المتعارف عليه أن تحمل حزم RPM اسماً مثل:

package-version-rev.arq.rpm

حيث package اسم البرنامج، و version رقم الإصدار، وتشير rev عادة إلى رقم المراجعة لحزمة RPM، والتي تعبر عن عدد المرات التي بُنِيَت فيها الحزمة، وتشير arq إلى المعمارية التي صمّمت لها الحزمة، سواء معماريات i386, i586, i686, x86_64, em64t, ia64 من إنتل و AMD، أو غيرها مثل alpha, sparc, ppc ...، أما معمارية noarch فتستخدم عادة عندما تكون الحزمة مستقلة عن العتاد، كأن تكون مثلاً مجموعة من النصوص البرمجية، أو src لتشير إلى حزم المصادر البرمجية. يتضمن التنفيذ الاعتيادي تشغيل rpm، وخيارات العملية التي ستُنفَّذ، مع اسم حزمة أو أكثر ليتم معالجتها معاً.

العمليات الاعتيادية مع حزم RPM تشمل:

1. معلومات الحزمة: يتم الاستعلام عن معلومات معينة عن الحزمة باستخدام الخيار q- مع اسم الحزمة (مع p- إذا كان ملف rpm). إذا لم تكن الحزمة مثبتة بعد، فسيكون الخيار q- مصاحباً خيار المعلومات المطلوبة، وإذا كان يفترض الاستعلام عن كل الحزم المثبتة في نفس الوقت ، فسيكون الخيار qa-. فمثلاً، للاستعلام عن حزمة مثبتة:

الطلب	خيارات RPM	النتائج
الملفات	rpm -ql	قائمة الملفات التي تحويها الحزمة.
المعلومات	rpm -qi	وصف الحزمة.
المتطلبات	rpm -qR	المتطلبات المسبقة، مكتبات أو برمجيات.

2. التثبيت: ببساطة، rpm -i package.rpm، أو برابط للحزمة؛ للتثبيت من مسارات http و ftp، سنحتاج فقط لاستخدام السياق //:http أو //:ftp للحصول على عنوان الحزمة. يمكن إكمال التثبيت في حال وجود اعتماديات الحزمة، سواء كانت مكتبات أو برمجيات يفترض أن تكون مثبتة مسبقاً. في حال عدم إكمال هذا المتطلّب، فسيقوم rpm بإخبارنا بالبرمجية الناقصة، واسم الحزمة التي توفرها. يمكننا إجبار rpm على تثبيت الحزمة (لكن البرمجية المثبتة قد لا تعمل في هذه الحالة) بالخيار force-- أو nodeps--، أو ببساطة بتجاهل المعلومات عن الاعتماديات.

إن عملية تثبيت حزمة (عبر rpm) تتضمن عمليات فرعية عديدة: أ. تفحص الاعتماديات إن وجدت؛ ب. تفحص التعارضات مع حزم أخرى مثبتة مسبقاً؛ القيام بمهام ما قبل التثبيت؛ ج. تقرير ما يجب عمله بملفات الإعدادات المرتبطة بالحزمة إذا كانت الموجودة مسبقاً؛ د. إلغاء تحزيم الملفات ووضعها في المكان الصحيح؛ هـ. تنفيذ عمليات ما بعد التثبيت؛ وفي النهاية، و. تسجيل تقرير بالمهام التي تم القيام بها في قاعدة بيانات RPM.

3. التحديث: مكافئ للتثبيت، لكنه يفحص أولاً إذا كان البرنامج مثبتاً مسبقاً، والأمر المستخدم هو rpm -U package.rpm، وستقوم بحذف التثبيت السابق تلقائيًّا.

4. التحقق: ستتغير كثير من الملفات أثناء العمل الاعتيادي للنظام. لهذا يسمح لنا rpm بفحص الملفات للكشف عن أية تغييرات ناجمة عن عملية اعتياديّة أو خطأ ما يمكن أن يسبب تلفاً في البيانات. بالأمر rpm -V package نتحقق من حزمة معينة، أما الأمر rpm -Va فسيتحقق من كلّ الحزم.

5. الحذف: إزالة الحزمة من نظام RPM (بالخيار e- أو erase--)؛ إذا كانت هناك اعتماديات، فعلينا إزالة الحزم التي تعتمد عليها أولاً.

مثال

في حالة التثبيت أو التحديث عن بعد، سيسمح لنا الأمر

rpm -i ftp://site/directory/package.rpm

بتنزيل الحزمة من مسار ftp أو وب المُعطى والاستمرار في هذه الحالة إلى عملية التثبيت.

علينا التحكم بأماكن الحصول على الحزم، وأن نستخدم فقط مصادر الحزم الموثوقة، كمنتج التوزيعة نفسه، أو المواقع

الموثوقة. عادة ما يتاح مع الحزمة توقيع رقمي لها، مما يمكننا من فحص موثوقيّتها. تُستَخدَم بصمات[3] md5 عادة للتأكد من أن الحزمة لم تتبدل، وأنظمة أخرى – مثل GPG (وهي إصدار جنو من PGP) – للتأكد من موثوقية مُنتج الحزمة. وبشكل مشابه، يمكننا أن نجد حزم RPM مختلفة مخزّنة على الإنترنت، حيث تتوفر لتوزيعات أخرى تستخدم أو تسمح باستخدام صيغة RPM.

لاستخدام آمن للحُزَم، توقّع المستودعات الرسمية وبعض المستودعات الخارجية هذه الأيام حزمها، باستخدام GPG التي سبق ذكرها مثلاً، يساعدنا هذا على التأكد (إذا كانت لدينا التواقيع) من أن الحزم تأتي من مصدر يعتمد عليه. عادة، سيضمِّن كل مزوّد (المستودع) بعض ملفات تواقيع PGP مع مفتاح لموقعه. عادة ما تكون هذه التواقيع مثبتة مسبقاً للمستودعات الرسميّة، لكن إذا كانت لمستودعات خارجية، فسنحتاج للحصول على ملف المفاتيح وتضمينه في RPM، عادة بهذا الأمر:

```
$ rpm --import GPG-KEY-FILE
```

على أن يكون GPG-KEY-FILE هو ملف GPG أو مسار الملف على الإنترنت، ، وعادة ما يكون لهذا الملف أيضاً مجموع md5 للتحقق من صحته. ويمكننا إيجاد المفتاح في النظام بالأمر:

```
$ rpm -qa | grep ^gpg-pubkey
```

يمكننا رؤية المزيد من المعلومات حول المفتاح الذي تم الحصول عليه بالأمر:

```
$ rpm -qi gpg-key-xxxxx-yyyyy
```

لحزمة RPM محدّدة، سنتمكن من فحص ما إذا كان لها توقيع، ومع أيّ ملف مفاتيح استُخدِمَت:

```
$ rpm --check-sig -v <package>.rpm
```

وللتأكد من أن حزمة ما صحيح بناء على التوقيع المتاح، يمكننا استخدام الأمر:

```
$ rpm -K <package.>.rpm
```

نحتاج لأن نكون حذرين بحيث نستورد المفاتيح فقط من المواقع التي نثق بها. عندما يجد RPM حزماً بتوقيع غير متوفر في نظامنا، أو عندما لا تكون الحزمة موقّعة فسيخبرنا، وعندها سيكون علينا أن نقرر ماذا نحن فاعلون.

---

3 يفضل البعض أن يطلق عليها "أكواد تحقق" أو "هاشات" كتعريب لكلمة hash، شخصيًّا أرى أن كلمة "بصمة" هي الأقرب للمعنى والأسهل للفهم في هذا السياق.

فيما يخص دعم RPM في التوزيعات، في فيدورا (وردهات ومشتقاتهما) RPM هي صيغة الحزم المبدئيّة والمستخدمة بشكل كبير في التوزيعة للتحديثات وتثبيت البرمجيات. تستخدم دبيان الصيغة المسماة DEB (كما سنرى)، هناك دعم لحزم RPM (الأمر rpm موجود)، لكنه فقط لفحص واستخراج معلومات الحزم. إذا كان من الضروري تثبيت حزمة RPM في دبيان، ننصح باستخدام أداة alien التي يمكنها تحويل صيغ الحُزَم - في هذه الحالة من RPM إلى DEB - والاستمرار للتثبيت من الحزمة المحوَّلة.

إضافة إلى نظام التحزيم الأساسيّ للتوزيعة، تدعم كل توزيعة هذه الأيام نظام إدارة برمجيات وسيط ذي مستوى أعلى يضيف طبقة فوقيّة للنظام الأساسيّ، ويساعد في مهام إدارة البرمجيات، ويضيف عدداً من الادوات لتحسين إدارة العمليّة.

في حالة فيدورا (ردهات ومشتقاتها)، يُستخدَم نظام YUM الذي يسمح لأداة ذات مستوى أعلى بإدارة وتثبيت الحزم في أنظمة RPM، إضافة إلى الإدارة الآليّة للاعتماديات بين الحزم. يسمح YUM بالوصول إلى مستودعات عديدة مختلفة، ويجعل إعداداتها في ملف مركزي (وعادة يكون /etc/yum.conf)[4]، وله واجهة أوامر بسيطة.

يعتمد إعداد YUM على:

ملف الخيارات	/etc/yum.conf [5]
مجلد لبعض الأدوات المرتبطة به	/etc/yum
مجلد لتحديد المستودعات (ملف لكلّ منها)، يتضمّن معلومات الوصول ومسار تواقيع gpg.	/etc/yum.repos.d

ملخص عمليات yum الاعتيادية:

الوصف	الأمر
تثبيت الحزمة ذات الاسم المُعطى	Yum install <name>
تحديث حزمة مثبتة	Yum update <name>

---

4 عادة ما يكون لـ YUM في التوزيعات الحديثة مجلد مساره /etc/yum.repos.d/ يحتفظ فيه بملفات إعداد المستودعات، وعادة ما يكون لكل مستودع ملف واحد أو أكثر، ويمكن وضع أكثر من مستودع في ملف واحد. أما /etc/yum.conf فصار يُستخدَم لضبط yum ذاته فقط، ولم يعد يُستَخدَم للاحتفاظ بمعلومات المستودعات هذه الأيام.

5 يذكر الكتاب الأصلي أن المسار هو /etc/yum.config، والصحيح هو أنه /etc/yum.conf، على الأقل في الإصدار 11 وما يليه من توزيعة فيدورا.

Yum remove <name>	التخلص من حزمة
Yum list <name>	البحث في أسماء الحزم (أسماء الحزم فقط!)
Yum search <name>	بحث مفصّل أكثر
Yum provides <name>	البحث عن الحزم التي تقدم الملف
Yum update	تحديث النظام بأكمله
Yum upgrade	كسابقه، ويشمل حزماً إضافيّة

وأخيراً، تقدّم فيدورا أيضاً بضع أدوات رسومية لـ YUM، مثل pup للتحكم بالتحديثات الأخيرة المتاحة، و حزمة إدارة البرمجيات pirutas. هناك أيضاً غيرها، مثل yumex بتحكم أقوى بإعدادات yum الداخلية.

5.3 دبيان: حزم DEB

لدبيان أدوات تفاعلية مثل tasksel الذي يجعل من الممكن اختيار مجموعات فرعية من الحزم مجمّعة ضمن أنواع المهام: حزم لـ X، وأخرى للتطوير، وأخرى للوثائق، وغيرها، أو مثل dselect التي تسمح لنا بتصفح قائمة كاملة للحزم الموجودة (هناك الآلاف منها) واختيار ما نريد تثبيته أو إزالته منها. في الحقيقة، هذه فقط واجهة لمدير البرمجيات ذي المستوى المتوسط APT.

على مستوى سطر الأوامر، هناك dpkg وهو الأمر ذو المستوى الأدنى (يمكن اعتباره مكافئاً لـ rpm)، لإدارة حزم برمجيات DEB مباشرة، عادة بالأمر dpkg -i package.deb للقيام بالتثبيت. كل أنواع المهام المرتبطة بالمعلومات، أو التثبيت، أو الإزالة، أو عمل تغييرات داخلية لحزمة البرمجيات يمكن تنفيذها.

المستوى المتوسط (كما في حالة YUM في فيدورا) يمكن تقديمها على أنها ادوات APT (معظمها أوامر apt-xxx). يسمح لنا APT بإدارة الحزم من قائمة الحزم الحالية والمتاحة اعتماداً على العديد من مصادر البرمجيات، سواء أقراص التثبيت نفسها، أومواقع وِب (HTTP) أو FTP. تتم هذه الإدارة مباشرة، بطريقة تجعل النظام مستقلاً عن مصادر البرمجيات.

يتم ضبط نظام APT من الملفات المتاحة في /etc/apt/، حيث /etc/apt/sources.list قائمة بالمصادر المتاحة؛ من الأمثلة عليها:

```
deb http://http.us.debian.org/debian stable main contrib non-free
debsrc http://http.us.debian.org/debian stable main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free
```

حيث العديد من المصادر "الرسميّة" لدبيان مجمّعة (وهي في هذه الحالة دبيان المستقرّة)، ويمكننا الحصول على حزم البرمجيات منها إضافة إلى تحديثات هذه الحزم. بالأساس نحدد نوع المصدر (وهي في هذه الحالة web/FTP) وهو الموقع، وإصدار التوزيعة (المستقر في هذا المثال) وتصنيفات البرمجيات التي نبحث عنها (حرّة، أو مساهمات أطراف خارجية، أو غير حرّة، أو تراخيص تجارية).

حزم البرمجيات متاحة لتوزيعات دبيان المختلفة، فهناك حزم للإصدار المستقر والاختبارية وغير المستقر. استخدام أحد هذه المصادر أو غيره يحدد نوع التوزيعة (بعد تغيير مصادر المستودعات في sources.list). يمكن الحصول على مصادر

برمجيات مختلطة، لكن لا ينصح بهذا، لأن التعارضات قد تنشأ بين إصدارات التوزيعات المختلفة.

ما إن ننتهي من ضبط مصادر البرمجيات، فالأداة الأساسية للتعامل معها على نظامنا هي apt-get، التي تسمح لنا بالتثبيت أو التحديث أو الإزالة من الحزم المنفردة وحتى تحديث التوزيعة بأكملها. هناك أيضاً واجهة لـ apt-get تُدعى aptitude والتي تطابق واجهة خياراتها apt-get عمليًّا (في الحقيقة، يمكن اعتبارها محاكياً لـ apt-get، حيث الواجهة مكافئة له)؛ من فوائد aptitude هو أنه يدير اعتماديات الحزم بشكل أفضل، ويتيح واجهة تفاعلية. ونرجو أن يصبح aptitude الواجهة الافتراضية في سطر الأوامر لإدارة الحزم في دبيان.

بعض الوظائف الأساسيّة في apt-get:

1. تثبيت حزمة معينة:

```
apt-get install package
```

2. إزالة حزمة:

```
apt-get remove package
```

3. تحديث قائمة الحزم المتاحة:

```
apt-get update
```

4. لتحديث التوزيعة، يمكننا استخدام هذه الخطوات الثلاثة على التوالي:

```
apt-get update
apt-get upgrade
apt-get list-upgrade
```

خلال هذه العملية الأخيرة، يمكننا إبقاء توزيعتنا محدثة دائماً، بتحديث الحزم المثبتة والتحقق من الاعتماديات مع الحزم الجديدة. من الأدوات المفيدة لبناء هذه القائمة apt-spy التي تحاول البحث عن أسرع المواقع الرسمية، أو netselect التي تسمح لنا باختبار قائمة المواقع. وهناك أمر آخر، يمكننا البحث عن المواقع الرسمية (يمكننا ضبطها بـ apt-setup) أو نسخ ملف مصادر موجود. البرامج الإضافية (من الأطراف الخارجية) قد تحتاج لإضافة مصادر إضافية أخرى (إلى /etc/sources.list)؛ يمكن الحصول على قوائم المصادر المتاحة (مثلاً: http://www.apt-get.org).

تحديث النظام تحديداً ينتج عنه تنزيل عدد كبير من الحزم (خاصة في الإصدار غير المستقر)، مما يجعل من المنصوح

به تفريغ الـ cache (المستودع المحلي) والحزم المنزّلة (يُحتفَظ بها في /var/cache/apt/archive/) والتي لن تُستخدَم بعد الآن سواء بالأمر apt-get clean للتخلص منها كلها، أو بالأمر apt-get autoclean للتخلص من الحزم غير المطلوبة، لأن هناك إصدارات جديدة بالفعل، ومبدئيًّا، لأننا لن نحتاجها مستقبلاً. علينا أن نفكر فيما إذا كنا سنحتاج هذه الحزم مرة أخرى لأغراض إعادة تثبيتها، حيث سنضطر لتنزيلها مجدداً في هذه الحالة.

يسمح نظام APT أيضاً ما يعرف بـ SecureAPT وهو إدارة آمنة للحزم عبر التحقق من مجاميع md5 وتواقيع مصادر البرمجيات (من نوع GPG). إذا لم تكن التواقيع متوفرة عند التنزيل، فسيقدم apt-get تبليغاً وعمل قائمة بالحزم غير الموقعة، للسؤال عما إذا سيوقف تثبيتها أم لا، تاركاً الخيار للمدير. قائمة المصادر المعتمد عليها حالياً يمكن الحصول عليها بالأمر:

```
# apt-key list
```

توزع مفاتيح GPG الرسمية لدبيان عبر حزمة، ويمكننا تثبيتها كالتالي:

```
apt-get install debian-archive-keyring
```

بديهيًّا، باعتبار أن لدينا ملف sources.list مع المواقع الرسمية. من المأمول أنه افتراضيًّا (اعتماداً على إصدار دبيان) ستكون هذه المفاتيح مثبتة مسبقاً عند بدء النظام. للمواقع الأخرى غير الرسمية التي لا توفر المفاتيح في حزمة، لكننا نعتبرها جديرة بالثقة، فيمكننا استيراد مفتاحها بالحصول عليه من المستودع (سيكون علينا البحث عن مكان وجود المفتاح. ليس هناك معيار محدد، لكنه عادة على الصفحة الرئيسية للمستودع على الإنترنت). نستخدم apt-key add مع المفتاح لإضافته، أو يمكننا أيضاً:

```
# gpg --import file.key
gpg --export --armor XXXXXXXX | apt-key add -
```

مع كون X رقماً ست-عشريًّا ذو علاقة بالمفتاح (انظر إلى تعليمات المستودع حول الطريقة المستحسنة لاستيراد المفاتيح والبيانات الضرورية).

ومن الوظائف المهمة لنظام APT استخراج معلومات الحزمة، باستخدام أداة apt-cache التي تسمح لنا بالتفاعل مع قوائم حزم برمجيات دبيان.

مثال

في apt-cache أوامر تسمح لنا بالبحث عن معلومات عن الحزم، مثل:

1. البحث عن حزم باستخدام اسم غير كامل:

```
apt-cache search name
```

2. عرض وصف الحزمة:

```
apt-cache show package
```

3. ما الحزم التي تعتمد عليها:

```
apt-cache depends package
```

أدوات ووظائف APT أخرى مثيرة للاهتمام:

apt-show-versions: يخبرنا بالحزم التي من الممكن تحديثها (ولأي إصدارات، انظر الخيار u-).

ستحتاج الوظائف الأخرى الأكثر تحديداً لأن يتم عملها بالأداة من المستوى الأدنى، مثل dpkg. على سبيل المثال، الحصول على قائمة بملفات حزمة مثبتة معينة:

```
dpkg -L package
```

قائمة كاملة بالحزم بالأمر:

```
dpkg -l
```

البحث عن الحزمة التي يأتي منها عنصر معين (ملف مثلاً):

```
dpkg -S file
```

يعمل هذا مع الحزم المثبتة، يمكن لـ apt-file أيضاً البحث عن الحزم التي لم تثبّت بعد.

في النهاية، تستحق أيضاً بعض الأدوات الرسومية لـ APT بعض الذكر، ومنها synaptic، و gnome-apt لجنوم، و kpackage أو adept لكدي[6]، إضافة إلى ما سبق ذكره من الأدوات النصّيّة مثل aptitude و dselect.

استنتاج: يجدر بنا أن نذكر بأن نظام إدارة APT (إضافة إلى قاعدة dpkg) مرن وقوي جدًّا عندما يتعلق الأمر

---

6 أظن كل هذه الادوات انقرضت عدا synaptic الذي ما يزال مستخدماً خاصة في أوساط مستخدمي أبونتو، أما البقية فقد حلّ محلها -gnome packagekit و kpackagekit للواجهتين والمتشابهين جدًّا في طريقة الاستخدام، ويعملان على التوزيعات الدبيانية والردهاتية وربما غيرها أيضاً ...

بإدارة التحديثات، وهو نظام إدارة الحزم المستخدم في دبيان والتوزيعات المشتقة منها، مثل: أبونتو وكبونتو ونوبِكس و Linex

[وسبيلي] وغيرها.

6 أدوات إدارية عامة

في مجال الإدارة، يمكننا أيضاً أن نذكر بعض الأدوات، تلك المصممة للأغراض الإدارية عامّة. رغم أنه يصعب أن يبقي المرء معلوماته محدّثة عندما يتعلق الأمر بهذه الأدوات بسبب الخطط الحالية للتوزيعات بإصداراتها المختلفة والتي تتطور بسرعة. سنذكر بعض الأمثلة (رغم أنها قد لا تعمل بشكل كامل في وقت ما[7]):

1. Linuxconf: هذه أداة إدارة عامة تجمع جوانب عديدة على شكل واجهة قوائم نصية، والتي تطورت في إصداراتها الأخيرة إلى واجهة وِب، يمكن استخدامها عملياً مع أيّ توزيعة جنو/لينكس وتدعم تفاصيل عديدة متأصلة في كل من هذه التوزيعات (لسوء الحظ، لم تحدّث منذ مدّة [طويلة جدًّا][8]).
2. webmin: هذه أداة إداريّة أخرى بدأ تصوّرها كواجهة وِب، تعمل بمجموعة من الملحقات يمكن إضافتها لكل خدمة نحتاج لإدارتها، عادة يكون لها نماذج تحدد معاملات إعداد الخدمة، توفر أيضاً إمكانية (إذا تم تفعيلها) السماح بالإدارة عن بعد من أي جهاز عليه متصفّح.
3. أخرى، مثل cPanel و ISPConfig[9].

إضافة إلى ذلك، تتيح كل من بيئتي سطح المكتب جنوم وكدي مفهوم "لوحات التحكم" التي تسمح بإدارة النواحي المرئية للواجهات الرسومية، إضافة إلى معاملات بعض أجهزة النظام.[10]

---

7 يرجى الأخذ بعين الاعتبار أن الكتاب يعود لعام 2009، وأن بعض هذه الأدوات لم تعد مفيدة الآن بالفعل، لذا اقتضى التنويه.

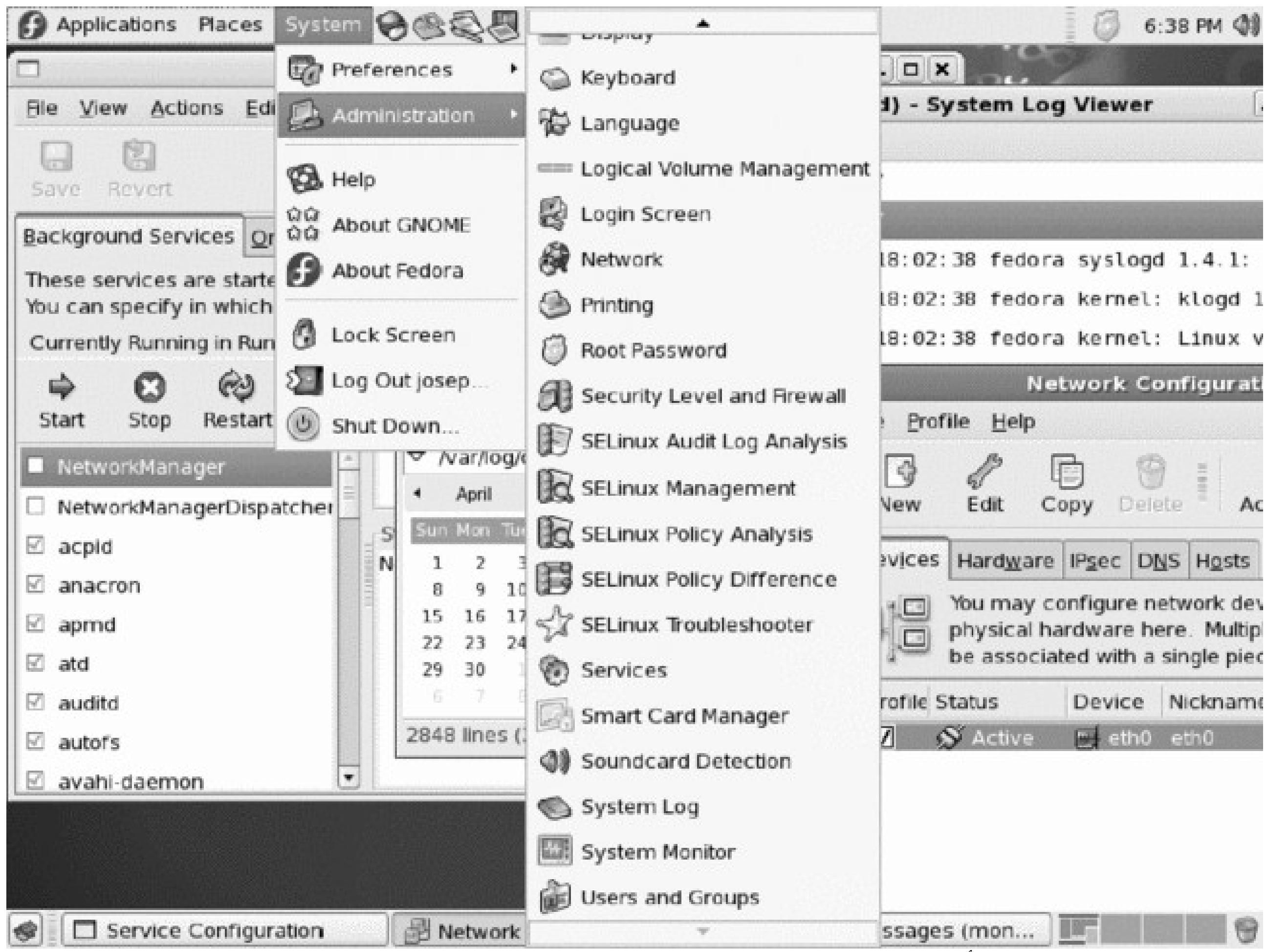
8 آخر تحديث له كان بتاريخ 18/1/2005 مما يجعله قديماً جدًّا وغير صالح للاستخدام هذه الأيام، خاصة مع التغيرات الكثيرة التي طرأت على أنظمة جنو/لينكس في نواحي عديدة!

9 كلتاهما متوفرتان الآن، وتعملان بشكل جيد جدًّا على لينكس، وتعد cPanel إحدى أكثر واجهات إدارة خوادم لينكس عبر المتصفح شعبية في العالم العربي، لكنها غير حرّة ولا مفتوحة المصدر ولا مجانية، تليها Webmin الحرّة ومفتوحة المصدر والتي تتوفر منها إصدارتان إحداهما مجانية وفيها كل ما يحتاجه المدير في الوضع الطبيعي، والأخرى مدفوعة وبها بعض المزايا الإضافيّة.

10 هناك أيضاً أدوات توفرها بعض التوزيعات مثل أدوات إدارة ماندريفا الرسومية، وأداة chkconfig في سطر أوامر ردهات، وهناك أدوات إدارية شائعة مثل system-config-xxxxx المبنية على مكتبة gtk الرسومية، وقد بدأتها ردهات وساهمت فيها نوفِل، وهي موجودة حاليًّا ضمن مشروع فيدورا ومعظم أجزائها معربة جيّداً. هناك غيرها لمن يبحث!

فيما يتعلق بالأدوات الإدارية الرسومية المنفردة، توفر بعض توزيعات جنو/لينكس بعضها بشكل مباشر (أدوات تصاحب جنوم وكدي)، وأدوات معدّة لإدارة الأجهزة (الطابعات، الأصوات، بطاقات الشبكة، وغيرها)، وأخرى لتنفيذ مهام معينة (اتصالات الإنترنت، ضبط بدء خدمات النظام، ضبط X Window، قراءة السجلات من الواجهة الرسومية، ...). كثير منها واجهات بسيطة للأدوات الأساسية في النظام، أو مطوّعة مزايا معيّنة في التوزيعة.

في هذا القسم، يجدر بنا أن نركّز على توزيعة فيدورا (ردهات ومشتقاتها) التي تحاول تقديم العديد من الأدوات (الأكثر بساطة) للوظائف الإدارية المختلفة، يمكننا إيجادها في سطح المكتب (في قائمة الإدارة)، أو في الأوامر مثل system-config-xxxxx لوظائف إدارية مختلفة، لكل من: الشاشة، الطابعة، الشبكة، الأمن، المستخدمين، الحزم، وغيرها. يمكننا رؤية بعضها في الشكل التالي:



شكل 3: بعض من الأدوات الإدارية الرسومية في فيدورا.

7 أدوات أخرى

لا يمكننا في المساحة المحدودة لهذه الوحدة أن نعلق على كل الأدوات التي يمكنها أن تقدم لنا منفعة كمدراء. سنقتبس بعض الأدوات التي يمكننا اعتبارها أساسية:

1. أوامر يونكس الأساسية العديدة: grep, awk, sed, find, diff, gzip, bzip2, cut, sort, df, du, cat, more, file, which ...
2. المحررات، وبشكل أساسي لأي مهمة تحرير، مثل: vi يستخدم كثيراً جدًّا في المهام الإدارية بسبب سرعة عمل تغييرات صغيرة على الملفات فيه. Vim هو المحرر المتوافق مع vi الذي يحويه نظام جنو/لينكس، يسمح بسياق ملون بلغات عديدة. Emacs هو محرر مكتمل جدًّا، ومطوّع ليلائم لغات برمجة مختلفة (أوضاع سياق وتحرير)، له بيئة كاملة جدًّا وإصدار رسومي يدعى Xemacs. وهناك أيضاً Joe، وهو محرر متوافق مع Wordstar، وغيرها الكثير ...
3. لغات البرمجة النصية، كأدوات إدارية، مثل: بيرل Perl المفيدة جداً في التعامل مع التعابير الاعتيادية وحليل الملفات (كالتصفية/الفلترة، والترتيب، وغيرها). PHP لغة مستخدمة بكثرة في بيئات الوِب. بايثون لغة أخرى يمكن عمل نماذج أولية للتطبيقات فيها ...
4. أدوات لتصريف وتصحيح اللغات عالية المستوى: GNU gcc (مصرّف سي و سي++)، و gdb (مصحح)، و xxgdb (واجهة X لأداة gdb)، و ddd (مصحح للعديد من اللغات).

الأنشطة

1. للقراءة السريعة، ألق نظرة على معيار FHS الذي سيساعدنا في الحصول على دليل جيد للبحث عن الملفات في توزيعتنا.

2. لمراجعة وزيادة المفاهيم والبرمجة في البرمجة النصية للصدفة في باش، انظر إلى: [Coo] [Bas].

3. لحزم RPM، كيف يمكننا تنفيذ بعض المهام التالية؟

   • اعرف الحزمة التي ثبّتت أمراً معيناً.

   • احصل على وصف للحزمة التي ثبتت أمراً.

   • احذف حزمة لا تعرف اسمها الكامل.

   اعرض كل الملفات التي كانت موجودة في نفس الحزمة مع ملف معين.

4. نفّذ نفس المهام المذكورة أعلاه، لكن لحزم دبيان، باستخدام أدوات APT.

5. حدّث توزيعة دبيان (أو فيدورا).

6. ثبّت أداة إداريّة عامّة الأغراض - كأداة Linuxconf أو Webmin على سبيل المثال - على توزيعتك. ماذا تقدم لنا؟ هل نفهم المهام المنفّذة والأثر الذي تسببه؟

المراجع

مصادر أخرى للمراجع والمعلومات

[Coo] [Bas] توفر مقدمة واسعة (ومفاهيم متقدمة) في برمجة نصوص الصدفة باش، إضافة إلى أمثلة عديدة.

يناقش [Qui01] صدفات البرمجة المختلفة في جنو/لينكس، إضافة إلى تشابهاتها واختلافاتها.

[Bai03] [Deb02] توفر رؤية واسعة لنظامي حزم البرمجيات لتوزيعتي دبيان وفيدورا/ردهات.

[Stu] مقدمة واسعة إلى الأدوات المتوفرة في جنو/لينكس.

نواة لينكس

د. جُسِــــپ جُبرا إستيڤَ

مقدمة

نواة نظام جنو/لينكس (المسمى عادة لينكس) هي القلب النابض للنظام: مسؤولة عن إقلاع النظام، وإدارة موارد الجهاز، بإدارة الذاكرة، ونظام الملفات، والإدخال والإخراج، والعمليات، والتخاطب المتبادل بين العمليات.

يعود تاريخها إلى عام 1991، عندما أعلن طالب فنلندي اسمه لينوس تورفالدز على قائمة أخبار أنه أنشأ لُبَّ نظام التشغيل الخاص به والذي عمل سوياً مع برمجيات مشروع جنو وأنه كان يعرضه لمجتمع المطورين لاختباره واقتراح تحسينات لجعله أكثر قابلية للاستخدام. كان هذا أصل نواة نظام التشغيل التي صارت تعرف لاحقاً بالاسم لينكس.

من المزايا الخاصة بلينكس أنه يتبع فلسفة البرمجيات الحرّة، فهو يوفر المصدر البرمجي لنظام التشغيل نفسه (النواة) بطريقة تجعله مثالياً للتعليم في مجال أنظمة التشغيل.

وهناك ميزة أخرى رئيسية، وهي أنه بالحصول على المصدر البرمجي، يمكننا تصريفه ليتكيف بشكل أفضل مع نظامنا، ويمكننا أيضاً ضبط معاملاتِه لتحسين أداء النظام.

في هذه الوحدة، سنرى كيفية التعامل مع عملية تحضير النواة لتناسب نظامنا. وكيف يمكننا بدأً من المصدر البرمجي أن نحصل على إصدار جديد للنواة مطوَّع لنظامنا. وبشكل مشابه، سنناقش طريقة تطوير الإعدادات والتصريف المتتالي وكيفية اختبار النواة الجديدة التي حصلنا عليها.

1 نواة نظام جنو/لينكس

اللُبّ أو النواة هي الجزء الأساسي لأي نظام تشغيل، حيث يمكن أن تتواجد أكواد الخدمات الأساسية للتحكم بالنظام بأكمله. بالأساس، يمكن أن تقسَّم بنيتها إلى سلسلة من عناصر الإدارة المصممة لما يلي:

1. عمليات الإدارة: ما المهام التي ستعمل، وبأي ترتيب، وبأيّ أولوية. ومن النواحي المهمة جدولة المعالج: كيف يمكننا أن نحسن وقت المعالج لتشغيل المهام بأفضل أداء ممكن أو بأكثر تفاعلية مع المستخدم؟
2. التواصل المتبادل بين العمليات والتزامُن: كيف تتخاطب المهام مع بعضها، مع أيّ آليات مختلفة، وكيف يمكن مزامنة مجموعات من المهام؟
3. الإدخال والإخراج (I/O): التحكم بالطرفيات وإدارة المصادر المرتبطة.
4. إدارة الذاكرة: تحسين استخدام الذاكرة، وخدمة ترحيل الصفحات، والذاكرة الافتراضية.
5. إدارة الملفات: كيف يتحكم النظام بالملفات المعروضة في النظام وينظمها وكيف يصل إليها.

شكل 1: المهام الأساسية للنواة مع الأخذ بالاعتبار التطبيقات والأوامر المنفذة

في الأنظمة المملوكة، تكون النواة "مخفية" بشكل كامل تحت طبقات برمجيات نظام التشغيل؛ لا يملك المستخدم نظرة واضحة بحقيقة النواة وليس لديه أيّ إمكانية لتعديلها أو تحسينها، سوى عبر استخدام المحررات الخفيّة لـ"سجلات" الداخلية، أو عبر برمجيات خاصة من أطراف خارجية عادة تكون غالية جدًّا. إضافة إلى هذا، فالنواة عادة ما تكون وحيدة، فهي النواة التي يوفّرها المصنّع الذي يحتفظ بحقّ تقديم أيّ تعديلات يريدها متى أراد، والتعامل مع الأخطاء التي تظهر في أوقات غير محددة عبر تحديثات تقدم لنا على أنها "تراقيع" لأخطاء (أو حزم خدمات).

ومن المشاكل الرئيسية المتعلقة بهذا التوجه هي تحديداً تَوفُّر هذه الترقيعات، فالحصول على تحديثات إصلاح الأخطاء أمر حرج، وإذا كانت هذه التحديثات متعلقة بالأمن، صار الأمر أكثر خطورة، لأنه لا يمكننا ضمان أمن النظام من المشاكل المعروفة حتى يتم إصلاحها. العديد من المؤسسات، والشركات الكبيرة، والحكومات، والمؤسسات العلمية والعسكرية لا يمكنها الاعتماد على مزاج المصنّع لإصلاح المشاكلة المتعلقة بتطبيقاتهم الحسّاسة.

تقدّم نواة لينكس حلًّا مفتوح المصدر مع الصلاحيات التي تتبع ذلك والتي تمكِّن من تعديل وتصحيح وإنشاء إصدارات جديدة وتحديثات بسرعة كبيرة على يد أيّ كان أينما كان إذا توفرت لديه المعرفة اللازمة لعمل ذلك.

يسمح هذا للمستخدمين ذوي الأوضاع الحرجة بالتحكم بتطبيقاتهم والنظام نفسه بشكل أفضل، وتمنح إمكانية ضمّ الأنظمة إلى نظام تشغيل مفصّل خصّيصاً لهم ومضبوط ليوافق ذوق كلّ مستخدم على حدة، وبالتالي الحصول على نظام تشغيل مفتوح المصدر طوّره مجتمع مبرمجين ينسّقون فيما بينهم عبر الإنترنت، ويمكن الوصول إليه لأغراض تعليمية لأن له أكواد مفتوحة المصدر وتوثيق متوفر بكثرة، حتى إنتاج أنظمة جنو/لينكس مطوّعة لتناسب احتياجات فرد أو مجموعة معيّنة.

لأن المصدر مفتوح، يمكن إيجاد التحسينات والحلول على الفور، على عكس البرمجيات المملوكة، التي تجبرنا على انتظار تحديثات المصنّع. وأيضاً، يمكننا تخصيص النواة قدر ما نريد، وهو متطلب أساسي في التطبيقات عالية الأداء مثلاً، كالتطبيقات التي يعتبر الوقت فيها عاملاً حرجًا، أو الحلول المتعلقة بالأنظمة المضمّنة (مثل الأجهزة النقّالة).

بتتبع قليل من تاريخ النواة (على عُجالة): طوّرها في البداية طالب فنلنديّ اسمه لينوس تورفالدز – وذلك عام 1991 – بنية عمل إصدار مشابه لنظام Minix (إصدار من UNIX للحواسيب الشخصية) لمعالج Intel 386. كان الإصدار الرسميّ

الأول منه Linux 1.0 في شهر آذار/مارس (3) عام 1994 والتي تضمّنت فقط التنفيذ على معماريات i386 ودعمت الأجهزة ذات المعالجات الأحاديّة. نُشِر لينكس 1.2 في نفس الشهر من عام 1995، وكان أول إصدار يغطي معماريات مختلفة مثل ألفا وسبارك ومِبس[1]. في الشهر السادس[2] من عام 1996، أضاف لينكس 2.0 المزيد من المعماريات، وكان أول إصدار تضمّن دعم المعالجات المتعددة SMP. في لينكس 2.2 - الشهر الأول[3] من عام 1999 - ازدادت فوائد SMP كثيراً، وأضيفت متحكمات لكمّ كبير من العتاد. تضمّن الإصدار 2.4 - الذي صدر في الشهر الاول لعام 2001 - تحسينات في دعم SMP، ودعم معماريات جديدة، ومتحكمات USB، ودعمًا لأجهزة بطاقات PC (لبطاقات PCMCIA للأجهزة المحمولة)، ودعم جزئي لـ PnP (الوصل والتشغيل plug and play)، ودعم وسائط التخزين و RAID، وغيرها. في الفرع 2.6 للنواة (12/2003)، تحسّن دعم SMP بشكل ملحوظ، وقدّمت استجابة أفضل لنظام جدولة المعالج، واستخدام خطوط المعالجة في النواة، ودعماً أفضل لمعماريات 64بت، ودعم المحاكاة، وتطويع محسّن للأجهزة النقالة.

يهمنا في التطوير، حيث أن منشئ النواة هو لينوس تورفالدز عام 1991 (الإصدار 0.01)، أن لينوس استمر بمتابعتها، لكن حسب ما سمح به عمله، ومع بلوغ النواة (وتطورها) ساعده في متابعة الإصدارات المستقرة المختلفة معاونون مختلفون، مع استمرار لينوس (قدر الإمكان) في تطوير وتصريف المساهمات للإصدارات الأخيرة من تطوير النواة. لقد كان المعاونون الرئيسيون في هذه الإصدارات كالتالي:

1. 2.0 ديفيد واينهول David Weinehall
2. 2.2 آلان كُكس Alan Cox (والذي طوّر ونشر ترقيعات لمعظم الإصدارات أيضاً)
3. 2.4 مارسيلو توساتي Marcelo Tosatti
4. 2.6 أندرو مورتون Andrew Morton ولينوس تورفالدز.

---

1 Alpha و Sparc و Mips.

2 الشهر السادس في التقويم الميلادي: شهر حزيران/يونيو/جوان.

3 الشهر الاول في التقويم الميلادي: كانون الثاني، يناير، جانفي.

ولنفهم القليل عن تعقيد نواة لينكس، لننظر إلى الجدول الذي يحوي قليلاً من التاريخ الملخّص لإصداراتها المختلفة، وحجمها من ناحية المصدر البرمجي. يُظهِر الجدول الإصدارات الإنتاجية فقط، يحدَّد الحجم (المقدَّر) بآلاف السطور من المصدر البرمجي:

الإصدار	تاريخ النشر	السطور البرمجية (بالآلاف)
0.01	1991 – 09	10
1.0	1994 – 03	176
1.20	1995 – 03	311
2.0	1996 – 06	649
2.2	1999 – 01	1800
2.4	2001 – 01	3378
2.6	2003 – 12	5930

كما نرى، لقد تطور النظام من ما يقارب عشرة آلاف سطر إلى حوالي ستة ملايين نهاية عام 2003.[4]

أما الآن، فالتطوير مستمر في الإصدار الثالث من النواة[5]، والذي تتضمنه معظم التوزيعات كإصدار مبدئيّ (ما زال بعضها يتضمّن الإصدارات الفرعية من 2,6)، رغم هذا فوجود قدر معين من المعرفة بالإصدارات السابقة ضروري، لأنه من الممكن أن نجد أجهزة بتوزيعات قديمة لم تُحدَّث، والتي قد نضطر لصيانتها أو ترحيلها إلى إصدارات أحدث.

لقد تسارع العمل على النواة أثناء تطوير على الفرع 2.6 بشكل ملحوظ، وذلك لأنّ كلًّا من لينوس تورفالدز وأندرو مورتون (المسؤولان عن الإصدار 2.6) انضمّا (عام 2003) إلى مختبر تطوير البرمجيات المفتوحة Open Source

---

4 لقد تخطّت النواة – حسب سلاشدوت وويكيبيديا – عشرة ملايين سطر برمجي عام 2008 شاملة التعليقات، وفي بداية عام 2012 (الإصدار 3.2 في شهر كانون الثاني/يناير) كسرت حاجز الخمسة عشرة مليون سطر – حسب ويكيبيديا الإنجليزية وموقع tom's hardware المتخصص بالعتاد – بنسبة زيادة 50% على حجمها في الشهور الـ 39 السابقة لها، وبزيادة قدرها خمسة ملايين سطر خلال ثلاث سنوات وثلاثة أشهر.

5 الإصدار المستقر الحاليّ هو الإصدار الثالث والإصدارات الفرعية عنه. تستخدم التوزيعات في الوقت الحالي فروع الإصدار الثالث، لكن هناك بعض الإصدارات القديمة من بعض التوزيعات والتي ما زالت مستخدمة للآن تعتمد على الإصدارات الفرعية من 2.6، كتلك المخصصة للخوادم أو التي لم تحدّث منذ مدّة. يفضل التأكد من ذلك، فربما يكون الإصدار 2.6 قد اختفى من الاستخدام الحقيقي لحظة قراءتك لهذا الكتاب. يرجى الانتباه إلى أن التغييرات الجوهرية بين 2.4 و 2.6 أكبر من تلك التي بين الإصدارات الأخيرة من 2.6 والإصدار 3.

Development Laboratory – OSDL، وهو تجمّع شركات متخصّصة في ترويج استخدام المصدر المفتوح وجنو/لينكس في الشركات (يتضمّن التجمّع من بين الشركات الكثيرة المهتمة بجنو/لينكس: ,HP, IBM, Sun, Intel, Fujitsu, Hitachi Toshiba, Red Hat, SuSE, Transmeta ...). نحن مقبلون الآن على موقف هامّ، حيث دَعَمَ تجمّع OSDL عمل كلّ من متابع (أندرو) ومطوّر (لينوس) الإصدار المستقر 2.6 للنواة، ليعملا بدوام كامل على الإصدارات والقضايا ذات العلاقة. بقي لينوس يعمل على النواة باستقلاليّة، بينما ذهب أندرو للعمل عند جوجل، حيث استمر في التطوير بدوام كامل، بعمل تراقيع مع مساهمات مختلفة للنواة. وفي وقت لاحق، صار OSDL منظّمة لينكس The Linux Foundation.

علينا أن نبقي في بالنا أنّه مع الإصدارات الجديدة للنواة قد حققت درجة عالية من التطوير والنبوغ، مما يعني أن الوقت بين نشر الإصدارات صار أطول (لكن هذا لا يشمل المراجعات الجزئيّة).

وهناك عامل آخر علينا أخذه بعين الاعتبار، وهو عدد الناس الذين يعملون على تطوير النواة حاليًّا. في البداية كان هناك عدد ضئيل من ناس ذوي المعرفة الكاملة بالنواة بأكملها، بينما هناك هذه الأيام الكثير من الناس يعملون في تطويرها. يقدّر عدد المساهمين هذه الأيام بحوالي ألفي شخص، بمستويات مختلفة للمساهمات، لكن عدد المطورين الذين يعملون بجهود كبير عليها يقدّر بالعشرات.

علينا أن نأخذ بعين الاعتبار أيضاً أن لدى معظم المساهمين معرفة جزئيّة بالنواة، وأنهم لا يعملون كلّهم في نفس الوقت، ولا مساهماتهم متساوية بنوعيتها (بعضهم يصلحون الأخطاء البسيطة فقط)؛ بل فقط قليل من الناس (كالمشرفين الذين لديهم معرفة كاملة بالنواة). هذا يعني أن التطويرات تحتاج بعض الوقت لتحدث، وأن المساهمات تحتاج لتصحيح للتأكد من أنها لا تتعارض مع بعضها، وأن هناك خيارات يجب أن تُتَّخَذ بين المزايا المختلفة.

فيما يتعلق بإصدارات نواة لينكس، فعلينا أن نأخذ ما يلي بعين الاعتبار:

1. حتى الفرع 2.6 من النواة، كانت تصنَّف إصدارات نواة لينكس إلى تقسيم ذي سلسلتين: كانت تعرف إحداهما

بالإصدار "التجريبي" Experimental (حيث الرقم الثاني رقم فرديّ، مثل 1.3 وفروعها، و 2.1 وفرعها، و 2.5 وفروعها)، والأخرى الإصدار الإنتاجيّ production (السلسلة الزوجيّة، مثل 1.2، و 2.0، و 2.2، و 2.4، وغيرها). السلسلة التجريبية، حيث الإصدارات التي نُقِلَت بسرعة والتي كانت تُستخدَم لاختبار ما استجدَّ من المزايا، والخوارزميات، ومشغّلات العتاد، وغيرها. بسبب طبيعة الأنوية التجريبية، يمكن أن تتصرّف بشكل غير متوقّع، مما يسبب خسارة في البيانات، أو خللاً في الجهاز، إلخ. ولهذا، لم تكن هذه الإصدارات مناسبة للبيئات الإنتاجيّة، ما لم يكن هذا لاختبار ميزة معينة (مع تحمّل المخاطرة المتعلقة بذلك).

كانت الأنوية الإنتاجيّة أو المستقرّة (السلاسل الزوجيّة) ذات مزايا محدّدة جيّداً، وعدد قليل من الأخطاء المعروفة، ومع معرّفات عتاد مجرّبة ومختبرة. كانت تنشر على فترات أطول من الإصدارات التجريبيّة، وكانت هناك العديد من الإصدارات، بعضها أفضل من غيرها. كانت توزيعات جنو/لينكس مبنية عادة على نواة مستقرة مختارة بشكل محدّد، وليس بالضرورة أحدث نواة إنتاجيّة منشورة.

2. ترقيم نواة لينكس المستخدَم في الإصدار 2.6 وفروعه ما زال يحتفظ ببعض النواحي الأساسيّة: يحدّد الإصدار بأرقام X.Y.Z، حيث يشير X إلى الإصدار الرئيسي، والذي يعني تغيرات هامّة على النواة[6]؛ Y هو الرقم الثاني في الإصدار، وعادة يتضمن تحسّناً في أداء النواة: الرقم Y زوجيّ للإصدارات المستقرّة، وفرديّ للإصدارات التطويريّة أو الاختباريّة؛ أما Z فهو رقم إصدار البناء، ويشير إلى رقم المراجعة للإصدار X.Y، فيما يتعلق بالترقيع أو التصحيحات المعمولة. لا يقوم الموزّعون بتضمين آخر إصدار من النواة، بل الإصدار الذي اختبروه أكثر ويمكنهم التحقّق من أنه مستقر للبرمجيات والعناصر التي يضمّنونها في توزيعتهم. بناء على هذه الخطّة التقليديّة في الترقيم (المتّبعة في الإصدارات الفرعية من 2.4، وحتى الإصدارات الأولى من الفرع 2.6)، تمّ عمل تغييرات ليتكيّف هذا النمط مع حقيقة كون النواة (الفرع 2.6) صارت مستقرّة أكثر (بتثبيت المتغيرين في X.Y إلى 2.6)، وأن المراجعات تقلّ وتقلّ (وبالتالي يقلّ تغير الأرقام في الخانتين الاوليين من رقم الإصدار)، لكن التطوير مستمر بسرعة جنونيّة.

6 باستثناء الإصدار 3.0 الذي لم يشمل تغييرات كبيرة على آخر إصدار فرعي من 2.6، ولم يشكّل نقلة نوعيّة، وهو متوافق تماماً معه، بل كان ذلك نتيجة لطول مدّة تطوير الإصدار 2.6، ورغبة مطوري النواة بالانتقال إلى الرقم التالي.

في أحدث خطّة، هناك أربعة أرقام لتحديد التغييرات الطفيفة على Z أو الاحتمالات المختلفة للمراجعات (مع ترقيعات مختلفة مضافة). وبهذا فالإصدار المحدّد بأربعة أرقام هو الذي يعتبر مستقرّاً. وهناك خطط أخرى مستخدمة في الإصدارات الاختباريّة المختلفة (عادة لا يُنصَح بها للبيئات الإنتاجيّة)، مثل لاحقة rc- (التي تشير إلى الإصدار المرشّح release candidate)، أو mm- التي تشير إلى الأنوية التجريبيّة التي فيها اختبار لتقنيات إبداعيّة مختلفة، أو git- التي تعدّ نوعاً من الإصدارات snapshots اليوميّة لتطوير النواة. خطط الترقيم هذه تتغير باستمرار لتتكيف مع طريقة عمل مجتمع النواة واحتياجاتها لتسريع التطوير.

3. للحصول على آخر نواة منشورة، ستحتاج لزيارة ملف نواة لينكس (على http://www.kernel.org) أو إحدى مراياها. سيكون من الممكن أيضاً إيجاد بعض التراقيع للنواة الأصليّة، التي تصحّح الأخطاء المكتشفة بعد نشر النواة.

من المزايا التقنيّة لنواة لينكس والتي يجدر بنا إبرازها:

1. هي نواة من النوع الأحاديّ: هي بالأساس برنامج أُنشئ كوحدة، لكنه مقسم افتراضيّاً إلى عدّة عناصر منطقيّة.
2. تدعم تحميل/تنزيل أجزاء من النواة، وتعرف هذه الأجزاء بالوحدات modules، وهي خصائص للنواة أو مشغّلات عتاد.
3. خيوط النواة: يُستخدَم للعمل الداخليّ للنواة عدد من خيوط التنفيذ، التي يمكن ربطها ببرنامج للمستخدم أو بوظيفة داخليّة للنواة. لم يستخدَم هذا المفهوم في لينكس بشكل مكثف. قدَّمت مراجعات الفرع 2.6 وما يليها دعماً أفضل، وهناك جزء كبير من النواة يعمل باستخدام خيوط التنفيذ العديدة هذه.
4. دعم التطبيقات متعدّدة الخيوط: دعم تطبيقات المستخدم لتعدّد الخيوط، حيث العديد من النماذج من نوع العميل/الخادم تحتاج لخوادم قادرة على متابعة طلبات عديدة متزامنة، وإعطاء خيط تنفيذ لكلّ طلب أو مجموعة من

الطلبات. للينكس مكتبته الخاصة من الخيوط التي يمكن استخدامها في التطبيقات متعددة الخيوط، ومع التحسينات التي تتم على النواة، سمحت النواة أيضاً باستخدام أفضل لتنفيذ مكتبات عمل الخيوط لتطوير التطبيقات.

5. النواة من نوع غير متساهل: هذا يعني أنه بداخل النواة، نداءات النظام (في الوضع الإشرافيّ) لا يمكن إيقافها عندما تكون مهمّة نظام قيد التنفيذ، وعندما تنتهي، يتم استئناف المهمّة السابقة. ولهذا، فإن النواة أثناء النداء لا يمكن إيقافها لتنفيذ مهمّة أخرى. وعادة ما تكون الأنوية المتساهلة مستخدمة في الأنظمة التي تعمل في الوقت الحقيقيّ، والتي تحتاج للسماح بما ذُكِرَ للتعامل مع الأحداث الحرجة. هناك إصدارات خاصّة من نواة لينكس للوقت الحقيقيّ تسمح بهذا بتقديم بعض النقاط الثابتة التي يمكن التبديل فيها. وقد تطور هذا المفهوم بشكل خاصّ في الفرع 2.6 من النواة، مما يسمح في بعض الحالات بمقاطعة بعض مهام النواة التي يمكن استئنافها للتعامل مع غيرها، ومن ثمّ استئنافها لاحقاً. مفهوم النواة المتساهلة هذا يمكن أن يكون مفيداً أيضاً في المهام التفاعليّة، حيث أنه إذا تمّ عمل نداءات (أو استدعاءات) مكلفة[7] للنظام، فستتسبّب هذه النداءات بتأخير في التطبيقات التفاعليّة.

6. دعم تعدّد المعالجات، والمعروف بالمعالجة المتعدّدة المتزامنة symmetric multiprocessing – SMP. يشير هذا المفهوم إلى الأجهزة التي تحوي في أبسط حالاتها معالجين اثنين، وحتى 64 معالج مركزي. لقد صارت هذه القضية مرتبطة بشكل خاص بالمعماريات ذات النوى (أي أنوية المعالج) المتعددة، والتي تسمح بوجود نواتي معالج اثنتين أو أربعة أنوية أو أكثر في الأجهزة الموجهة للمستخدمين المنزليين. يمكن للينكس استخدام عدد من المعالجات بحيث يتعامل كلّ معالج مع مهمة واحدة أو أكثر. لكن بعض أجزاء النواة [قديماً] قلّلت من الأداء، حيث كانت تلك الأجزاء معدّة للعمل على معالج واحد، وأجبرت النظام بأكمله على التوقف في حالات معيّنة. إن SMP من أكثر التقنيات التي تدارسها مجتمع نواة لينكس، وقد تم إنجاز تحسينات هامة في الفرع 2.6. ولهذا يعتبر أداء SMP من العوامل المؤثرة على اتخاذ القرار عندما يتعلق الأمر بالشركات التي تتبنى لينكس كنظام تشغيل للخوادم.

7. أنظمة الملفات: للنواة معمارية نظام ملفات جيدة، والعمل الداخلي فيها مبني على نظرية نظام ملفات وهميّ VFS

7 من ناحية مقدار استخدام قدرات العتاد والمدّة المطلوبة لتنفيذها.

Virtual File System –، الذي يمكن تطويعه ليتوافق مع أيّ نظام حقيقيّ. ونتيجة لذلك، فإن لينكس على الأرجح هو نظام التشغيل الذي يدعم أكبر عدد من أنظمة الملفات، ومما يدعمه: ext2, MSDOS, VFAT, NTFS، والأنظمة السجلّيّة، مثل: ext3, ReiserFS, JFS (IBM), XFS (Silicon), NTFS, ISO9660 (CD), UDF والعديد من الأنظمة الأخرى التي أضيفت في المراجعات.

المزايا الأخرى غير التقنية (بعض الأمور التسويقية):

1. لينكس مجاني: بلينكس و برمجيات جنو الموجودين في أيّ توزيعة، يمكننا الحصول على نظام شبيه بيونكس بتكلفة العتاد فقط، ففيما يتعلق بتكلفة توزيعات جنو/لينكس، يمكننا الحصول عليها مجاناً. رغم هذا، فمن المنطقي أن تدفع مبلغاً إضافيًّا صغيراً لتوزيعة كاملة بمجموعة كاملة من أدلة الاستخدام ودعم فني بتكلفة كلية أقل [بكثير] مما قد تدفعه لقاء بعض الأنظمة المملوكة، أوبالمساهمة بذلك الشراء في تطوير التوزيعات التي نفضلها أو التي نجدها عملية اكثر.
2. لينكس يمكن تعديله: تسمح لنا رخصة GPL بقراءة وتعديل المصدر البرمجي للنواة (في حال توفر المعرفة المطلوبة بكيفية عمل ذلك).
3. يمكن للينكس العمل على الأجهزة القديمة المحدودة، فمثلاً، يمكن إنشاء خادم شبكة على جهاز 386 بذاكرة RAM قدرها 4MB (هناك توزيعات مخصصة للموارد المحدودة).
4. لينكس نظام قوي: الهدف الأساسي للينكس هو الجودة، فهو يهدف إلى الحصول على أفضل استخدام للعتاد المتاح.
5. جودة عالية: أنظمة جنو/لينكس مستقرة جدًّا، ولها معدل خطأ منخفض، وتقلل من الوقت اللازم لصيانة الأنظمة.
6. النواة صغيرة نسبيًّا ومضغوطة: يمكن وضعها مع برامج بسيطة على قرص ذي حجم 1.44MB فقط[8] (هناك عدد من التوزيعات التي تأتي على قرص مرن واحد مع برامج بسيطة).
7. لينكس متوافق مع عدد كبير من أنظمة التشغيل، ويمكنه عمليًّا قراءة أي نظام ملفات يمكن التخاطب عبره على

---

8 المقصود هنا بالقرص ذي السعة 1.44MB هو القرص المرن المربّع صغير الحجم المصنوع من البلاستيك. إذا كنت من مستخدمي الحاسوب في التسعينات فلربما تذكره.

الشبكة وذلك بتقديم أو استقبال خدمات من أيّ من هذه الأنظمة. ويمكنه أيضاً تشغيل برامج أنظمة أخرى أيضاً عبر مكتبات معينة (مثل برامج MSDOS و Windows و BSD و Xenix وغيرها) على معماريات x86.

8. للينكس دعم ممتد: إن ما يحظى به لينكس لا ينافسه فيه نظام آخر من ناحية سرعة وعدد التراقيع والتحديثات، ولا حتى الأنظمة المملوكة. وعند وجود مشكلة معينة، فإن هناك ما لا يعدّ من القوائم البريدية والمنتديات التي يمكنها المساعدة في حلّ أيّ مشكلة خلال بضع ساعات فقط. المشكلة الوحيدة تتعلق بمتحكمات العتاد الجديد الذي لا يزال بعض مصنعيه ممانعين لتوفير مشغلات له لغير الأنظمة المملوكة. لكن هذا يتغير باستمرار، وهناك عدد من أهم المصنعين في مجالات مثل بطاقات العرض (NVIDIA, ATI) والطابعات (Epson, HP) تقوم بتوفير متحكمات لأجهزتها بالفعل.

2 ضبط أو تحديث النواة

كمستخدمي أو مدراء أنظمة جنو/لينكس، علينا أن نبقي في بالنا الإمكانيات التي توفرها لنا النواة لتطويع متطلباتنا ومعداتنا.

في وقت التركيب، توفر توزيعات جنو/لينكس سلسلة من أنوية لينكس الثنائية المضبوطة والمصرفة مسبقاً، ويكون علينا في هذه الحالة اختيار أي الأنوية المتاحة متوافق أكثر مع عتادنا. هناك أنوية عامّة الأغراض generic، وأخرى معدة لأجهزة IDE، وأخرى لـ SCSI، وأخرى توفر خليطاً من متحكمات العتاد.

وهناك خيار آخر أثناء التثبيت وهو إصدار النواة. تستخدم التوزيعات عادة تراكيب يعتبرونها مختبرة جيداً ومستقرة بحيث لا تسبب أيّ مشكلات للمستخدمين. فمثلاً، العديد من التوزيعات تأتي مع إصدار النواة الثالث أو إصدار 2.6 كونه يعتبر الأكثر استقراراً[9] (في وقت إطلاق التوزيعة). وفي حالات معينة، وكبديل لها، فيمكن أن تتوفر إصدارات أحدث أثناء التثبيت، مع دعم محسّن لأجهزة أكثر حداثة (كأحدث أجيال بعض الأجهزة) والتي لم تكن قد اختُبِرَت كفاية في وقت نشر التوزيعة.

يقوم الموزّعون بتعديل النواة لتحسين أداء توزيعتهم أو لتصحيح أخطاء في النواة Hثناء الاختبارات. ومن التقنيات الشائعة أيضًا في التوزيعات المملوكة تعطيل المزايا المسببة للمشاكل والتي يمكن أن تتسبب بأخطاء للمستخدمين أو التي تتطلب ضبطاً معيناً للجهاز، أو عند اعتبار ميزة معينة غير مستقرة بشكل كافٍ ليتم تضمينها مبدئيًّا.

يقودنا هذا إلى اعتبار أنه بغض النظر عن مدى جودة أداء التوزيعة لوظيفة تطويع النواة توزيعتها، فإننا قد نواجه مشاكل معينة:

1. النواة غير محدثة لأحدث إصدار مستقر متاح؛ بعض الأجهزة الحديثة غير مدعومة.
2. النواة القياسية لا تدعم الأجهزة التي لدينا لأنها غير مفعلة فيها.
3. المتحكمات التي يقدمها لنا مصنّع ما تتطلب إصداراً حديثاً للنواة أو تعديلات معينة عليها.

---

9 الإصدار الثالث مستقر الآن، وهو الذي يأتي مع التوزيعات في الوقت الحالي وليس 2.6؛ باستثناء بعض الإصدارات القديمة.

4. وقد يكون العكس، فالنواة حديثة جدًّا، ولدينا عتاد قديم لم يعد مدعوماً في الأنوية الحديثة.
5. النواة كما هي لا تستخلص أفضل أداء من الأجهزة.
6. تتطلب بعض التطبيقات التي نرغب باستخدامها دعم نواة حديثة أو إحدى [أو بعض] مزاياها.
7. نريد بأن نكون في المقدمة ونحصل على أحدث إصدارات، ونتحمل مخاطرة بتثبيت أحدث إصدارات نواة لينكس.
8. نرغب بتحري أو اختبار التحسينات الجديدة في النواة، أو نرغب في تعديل النواة.
9. نريد برمجة مشغّل لجهاز غير مدعوم.
10. ...

لهذه الأسباب وغيرها، قد لا نكون سعداء بالنواة التي لدينا؛ وفي هذه الحالة لدينا خياران: تحديث النواة الثنائية للتوزيعة، أو تفصيل واحدة باستخدام المصدر.

فلنلقِ نظرة على بعض القضايا المتعلقة بالخيارات المختلفة وما يتعلق بها:

1. تحديث نواة التوزيعة: ينشر الموزع بالعادة أيضاً تحديثات للنواة عند صدورها. عندما ينشِئ مجتمع النواة إصداراً جديداً لها، يقوم كلّ موزع بضمها إلى توزيعته وعمل الاختبارات ذات العلاقة. فيما يلي مدة الاختبار، يتم تحديد الأخطاء المحتملة، ومن ثم يتم تصحيحها، يلي ذلك إصدار التحديثات اللازمة من النواة للتوزيعة. يمكن للمستخدمين تنزيل المراجعات الجديدة للتوزيعة من الموقع، أو تحديثها عبر نظام حزم آليّ آخر عبر مستودع للحزم. في العادة، يتم التأكد من مراجعة النظام، وتنزيل النواة الجديدة وعمل التغييرات اللازمة، بحيث يعمل النظام في المرة التالية بالنواة الجديدة، مع الاحتفاظ بالإصدارات القديمة لاستخدامها في حال وجود مشكلة ما.

   يبسّط لنا هذا النوع من التحديث العملية كثيراً، لكنه قد لا يحل مشاكلنا، حيث قد لا يكون عتادنا مدعوماً بعد، أو أن مزايا النواة التي تحتاج للاختبار لم تُتح بعد في الإصدار الذي لدينا من التوزيعة؛ سنحتاج لتذكر أنه ليس هناك سبب يدعو الموزعين لاستخدام أحدث إصدار متاح (في kernel.org مثلاً)، ولكن الإصدار الذي يعتبرونه مستقراً بالنسبة لتوزيعتهم.

إذا لم يكن عتادنا مفعلاً مبدئيًّا في النواة الجديدة أيضاً، فسنجد أنفسنا واقعين بنفس المشكلة. أو ببساطة، إذا كنا نريد أحدث إصدار، فهذه الطريقة غير مجدية.

2. تفصيل النواة (هذه الخطوة مشروحة بالتفصيل في الفصل التالي). في هذه الحالة، سنذهب إلى مصادر النواة ونضبط العتاد أو الخصائص المطلوبة "يدوياً". سنمر عبر عملية ضبط وتصريف المصدر البرمجي للنواة وذلك لإنشاء نواة ثنائية سنثبتها على النظام، وبهذا تكون تلك الخصائص متاحة في المرة التالية التي يقلع فيها النظام.

يمكننا هنا أن نرى أيضاً خيارين إضافيين، فإما أن نحصل على الإصدار "الرسميّ" للنواة مبدئيًّا (من kernel.org)، أو يمكننا الذهاب إلى المصادر التي توفرها التوزيعة نفسها. علينا أن نبقي في بالنا أن التوزيعات مثل فيدورا ودبيان تقوم بالكثير من العمل لتطويع النواة وتصحيح أخطائها التي تؤثر على توزيعتهم، مما يعني أنه في بعض الحالات يمكننا أن نحصل على تصحيحات إضافية للأكواد الإضافيّة للنواة. ومرة أخرى، المصادر التي توفرها التوزيعة قد لا تكون بالضرورة أحدث إصدار منشور للنواة.

> يسمح لنا هذا النظام بالحصول على أعلى موثوقيّة وتحكم، لكن بتكلفة إدارية عالية؛ حيث سنحتاج لأن تكون لدينا معرفة موسّعة بالأجهزة والخصائص التي نختارها (ماذا تعني، وما المزايا التي توفرها)، إضافة إلى العواقب التي قد تترتب على القرارات التي نتخذها.

3 عملية الإعداد والتصريف

ضبط النواة عملية مكلفة، وتتطلب وجود معرفة واسعة عند الشخص الذي سيقوم بالمهمة، وهي أيضاً إحدى المهام الحرجة التي يعتمد عليها استقرار النظام، آخذين بعين الاعتبار طبيعة النواة، وهي المكون الأساسيّ في النظام.

يمكن أن يسبب أي خطأ في العملية عدمَ استقرار أو خسارة النظام. ولهذا، ينصح بعمل نسخ احتياطي لبيانات المستخدمين والإعدادات التي قمنا بضبطها، أو – إذا كانت لدينا الأجهزة المطلوبة – عمل نسخ احتياطي للنظام بأكمله. ينصّح أيضاً بوجود قرص إقلاع (كالقرص الحيّ للتوزيعة) لمساعدتنا في حال حدوث أيّ مشكلة، أو قرص إنقاذ كالذي توفره معظم التوزيعات والذي يسمح لنا بعمله من قرص التوزيعة (أو بتوفير قرص إنقاذ جاهز للتوزيعة).[10]

دون مبالغة، إذا كنا نعلم ما نفعله، واتخذنا الاحتياطات اللازمة، فإن الأخطاء غالباً لا تظهر مطلقاً.

لنلقِ نظرة على العملية المطلوبة لتثبيت وإعداد نواة لينكس. في القسم التالي، سنرى:

1. حالة الإصدارات القديمة 2.4[11]
2. بعض الاعتبارات التي يجب اتخاذها عند الهجرة إلى 2.6
3. تفاصيل محدّدة تتعلق بالإصدار 2.6 من النواة.
4. حالة خاصة لتوزيعة دبيان، والتي لها طريقتها الخاصة الأكثر مرونة لتصريف النظام (طريقة دبيان).

لم تعد التوزيعات الحالية تقدّم الإصدارات 2.4 [بل إنها لم تعد توفر 2.6 أيضاً، بل الإصدار الثالث وفروعه]، لكن علينا الأخذ بعين الاعتبار أننا في أكثر من سياق قد نجد أنفسنا مضطرين لترحيل نظام معين إلى إصدار جديد أو صيانته وإبقاؤه على إصداره القديم، بسبب عدم توافقية مع الأجهزة القديمة الموجودة وغير المدعومة.[12]

---

10 يمكن أيضاً استخدام أحد أقراص أو إحدى توزيعات الصيانة، مثل Parted Magic أو غيرها لهذا الغرض.

11 قد تحوي ترجمة هذا الجزء تعديلات عديدة دون الإشارة إلى وجودها.

12 شخصيًّا، أستبعد أن تواجه مثل تلك الحالات هذه الأيام، لكنك ستحتاج لقراءتها لأخذ فكرة عامة عن كيفية التعامل مع النواة، ومن ثم ستتوضح لك الاختلافات مع الإصدار 2.6 وما يليه في الفصل الذي يلي ذلك.

المفهوم العام لعملية التصريف والإعداد سيتم شرحها في القسم الأول (المتعلق بالإصدار 2.4)، حيث أن معظمها عامّة، ومن ثم سنرى الاختلافات المتعلقة بالإصدارات الحديثة.

3.1 تصريف النواة للإصدارات 2.4

التعليمات لمعماريات Intel x86 تحديداً، وبالمستخدم الجذر (لكن بعض أجزاء هذه العملية يمكن أن تتم بالمستخدم العادي):

1. **الحصول على النواة:** يمكننا مثلاً زيارة الموقع http://www.kernel.org (أو أحد خوادم FTP التابعة له)، وتنزيل الإصدار الذي نرغب باختباره. هناك مرايا للدول المختلفة. في معظم توزيعات جنو/لينكس، مثل فيدورا/ردهات أو دبيان، يتوفر المصدر البرمجي للنواة كحزمة أيضاً (وعادة ما تحوي بعض التعديلات)، فإذا كنا نتعامل مع إصدار النواة الذي نحتاجه، فمن المفضل استخدامها (عبر حزمة مصدر النواة أو ما شابه). أما إذا كنا نريد أحدث الأنوية، فقد لا تكون متاحة في التوزيعة، وسيكون علينا الذهاب إلى الموقع الرسميّ http://www.kernel.org.

2. **استخراج النواة:** عادة ما تستخرج مصادر النواة وتوضع في المجلد /usr/src/، لكننا ننصح باستخدام مجلد منفصل حتى لا نخلطها مع ملفات المصدر التي قد تحويها التوزيعة. فمثلا، إذا كان المصدر يأتي في ملف مضغوط من نوع bzip2:

```
bzip2 -dc linux-2.4.0.tar.bz2 | tar xvf –
```

أما إذا كانت المصادر تأتي في ملف gz، فسنستبدل gzip مكان bzip2. وعندما نفكّ ضغط المصادر، فسننشئ مجلداً ذي اسم يبدأ بـ kernel وبعده رقم إصدار النواة والذي سندخله لنضبط إعدادات النواة.

قبل اتخاذ خطوات التصريف، علينا التأكد من أن لدينا الأدوات المناسبة، وخاصة مصرّف جنو gcc، وأداة make، وأدوات جنو الأخرى المكمّلة اللازمة للعملية. ومن الأمثلة على ذلك modutils، وهي مجموعة أدوات لاستخدام والتعامل مع الواحدات المتغيرة للنواة. وكذلك، علينا أيضاً أن نأخذ بعين الاعتبار – لخيارات الإعداد المختلفة – عدداً من المتطلبات المسبقة التي تأتي على شكل مكتبات مرتبطة بواجهات الإعداد المستخدمة (مثل ncurses لواجهة menuconfig [النصيّة]).

وبشكل عام، ننصح بتفقّد توثيق النواة (سواء عبر الحزمة أو في المجلد الرئيسي للمصدر) لمعرفة المتطلبات المسبقة

وإصدارات مصدر النواة التي نحتاجها للعملية. ننصح بدراسة ملفات README في المجلد الجذر للمصدر البرمجي للنواة، وكذلك Documentation/Changes أو فهرس توثيق النواة في Documentation/00-INDEX.

إذا كنا قد قمنا بعمل التصريفات السابقة في نفس المجلد، فسنحتاج للتأكد من أن المجلد الذي نستخدمه خالٍ من التصريفات السابقة، يمكننا تنظيفه بتنفيذ make mrproper (من المجلد الرئيسيّ للمصدر البرمجي).

لعملية ضبط النواة، لدينا عدد من الطرق البديلة، والتي توفر لنا واجهات مختلفة لضبط المعامِلات المختلفة للنواة (والتي تكون مخزنة في ملف إعدادات، وعادة ما يكون config. في المجلد الجذر للمصادر البرمجية). البدائل المختلفة هي:

- make config : من سطر الأوامر نُسأل عن كلّ خيار، ويطلب منا تأكيده عبر الحرفين y و n (وتشيران لكلمتي yes و no بالإنجليزيّة)، أو بكتابة الخيار، أو نُسأل عن القيم المطلوبة. أو الإعداد الطويل والذي نُسأل فيه العديد من الأسئلة، واعتماداً على كلّ إصدار، سيكون علينا كذلك الإجابة على مئات الأسئلة (أو ربما أكثر، وهذا يعتمد على الإصدار).
- make oldconfig : هذا الأمر مفيد إذا كنّا نرغب بإعادة استخدام الإعداد المستخدَم مسبقاً (والذي عادة ما يكون مخزّناً في الملف config. في المجلد الجذر للمصدر البرمجي)، وسنحتاج لأن نأخذ بعين الاعتبار أنه سيكون صالحاً فقط إذا كنّا سنصرّف نفس إصدار النواة، حيث أن الإصدارات المختلفة للنواة قد تحوي خيارات متغيرة.
- make menuconfig : إعداد معتمِد على قوائم نصّيّة، وهو مرضٍ ويمكن الاعتماد عليه؛ يمكننا أن نفعّل أو نعطّل ما نريد، وهو أسرع من make config.
- make xconfig : الأفضل بينها، ويعتمد على حوارات رسومية تعتمد على X Window. سنحتاج لوجود مكتبات tcl/tk مثبتة، حيث أنّ أداة الإعداد هذه مبرمجة بهذه اللغة. الإعداد مبني على جداول حوارات وأزرار/صناديق اختيار، ويمكن أن يتم الامر عبرها بسرعة، وبها مساعدة بتعليقات على معظم الخيارات. لكن فيها عيباً، وهو أن بعض الخيارات قد لا تظهر (هذا يعتمد على ما إذا كان برنامج الإعداد محدّثاً،

وأحياناً لا يكون كذلك). في هذه الحالة الاخيرة، فإن make config (أو make menuconfig) هي الأداة الوحيدة التي يمكننا أن نتأكد من أنها توفر لنا كلّ الخيارات التي يمكننا الاختيار فيما بينها، بالنسبة للأنواع الأخرى للإعداد، فهذا يعتمد على ما إذا كانت تلك البرامج مطوّعة للخيارات الجديدة عند إصدار النواة. رغم هذا، فعادة ما يحاولون عمل ذلك في نفس الوقت.

Linux Kernel Configuration

Code maturity level options	Fusion MPT device support	Sound
Loadable module support	IEEE 1394 (FireWire) support (EXPERIMENTAL)	USB support
Processor type and features	I2O device support	Additional device driver support
General setup	Network device support	Bluetooth support
Memory Technology Devices (MTD)	Amateur Radio support	Profiling support
Parallel port support	IrDA (infrared) support	Kernel hacking
Plug and Play configuration	ISDN subsystem	Library routines
Block devices	Old CD-ROM drivers (not SCSI, not IDE)	
Multi-device support (RAID and LVM)	Input core support	
Cryptography support (CryptoAPI)	Character devices	
Networking options	Multimedia devices	Save and Exit
Telephony Support	Crypto Hardware support	Quit Without Saving
ATA/IDE/MFM/RLL support	File systems	Load Configuration from File
SCSI support	Console drivers	Store Configuration to File

شكل 2: ضبط النواة (make xconfig) من الواجهة الرسومية في X Window

ما إن تتمّ عملية الإعداد، سنحتاج لحفظ الملف (config.)، حيث يحتاج الإعداد لوقت ليس بالقليل. وأيضاً، قد يكون من المفيد أيضاً أن يكون الإعداد جاهزاً إذا كانت الخطّة تقتضي عمل ذلك في العديد من الأجهزة المتشابهة أو المتماثلة.

وهناك أمر آخر مهم يتعلق بخيارات الإعداد، وهو أننا سنُسأل في العديد من الحالات عمّا إذا كنّا نريد خصائص محدّدة مدمجة في النواة أو كوحدة [منفصلة] (سنوفر المزيد من التفاصيل عنها في القسم المتعلق بالوحدات). هذا قرار مهم نوعاً ما، حيث أنه في بعض الحالات سيكون لدينا خيار يؤثر على أداء النواة (وبهذا على أداء النظام ككُل).

لقد صارت نواة لينكس ضخمة جدًّا، وذلك بسبب تعقيدها، وبسبب متحكّمات الأجهزة (المشغّلات أو التعاريف) التي تتضمّنها. إذا ضمّنّا كلّ شيء، فقد ننشئ ملف نواة ضخم جدًّا يحتل كمية كبيرة من

الذاكرة، مما يبطئ بعض النواحي الوظيفيّة. وحدات النواة طريقة تجعل من الممكن تقسيم جزء من النواة لأقسام أصغر يتم تحميلها بشكل متغير عند الطلب أو عند الحاجة لها سواء كتحميل فقط أو لاستخدام ميزة. الخيار العادي هو تضمين ما يعتبر أساسيًّا للعمل أو حرجاً للأداء في النواة، ولترك الأجزاء أو المتحكّمات التي ستستخدم بشكل متقطّع كوحات للإضافات المستقبلية للمعدّات.

- من الحالات الواضحة متحكّمات العتاد: إذا كنّا نحدّث الجهاز، فإننا عندما نأتي لإنشاء النواة، قد نكون غير متأكدين أي عتاد سيكون لدينا لاحقاً: فمثلاً، أي بطاقة شبكة سيكون لدينا؛ لكننا نعلم بأنه سيكون موصولاً بالشبكة، ولهذا، فسيكون دعم الشبكة مضمّناً في النواة، لكن بالنسبة لمتحكّمات العتاد، يمكننا اختيار بعض (أو كلّ) هذه المتحكّمات وتثبيتها كوحدات modules. ومن ثمّ سنتمكن من تحميل الوحدة المطلوبة عند حصولنا على البطاقة؛ سنستفيد من ذلك أيضاً إذا كنا نفكر بتغيير البطاقة لاحقاً، حيث لن نحتاج سوى لتغيير الوحدة التي يتمّ تحميلها . إذا كان لدينا وحدة واحدة فقط مضمّنة في النواة وغيرنا البطاقة فسنكون مضطرين لإعادة كلّ من ضبط وتصريف النواة مع متحكّم البطاقة الجديدة.
- والحالة الأخرى هي التي تظهر (رغم أنها غير شائعة) عندما يكون لدينا جهازان غير متوافقين مع بعضهما، أو عندما يكون أحدهما يعمل (فمثلاً، هذا يحصل مع طابعات المنفذ المتوازي والعتاد المتصل بالمنفذ المتوازي). ولهذا، في هذه الحالة، نحتاج لوضع المتحكمات كوحدات وتحميل أو تنزيل الوحدة التي نريدها.
- أنظمة الملفات مثال آخر على ذلك. عادة ما نرغب بأن يتمكن نظامنا من الوصول إلى عدد منها، مثل ext2 أو ext3 (الذين ينتميان للينكس)، أو VFAT (الذي ينتمي لوندوز ME/95/98)، وسنفعلهما في ضبط النواة. إذا كنّا سنحتاج لقراءة نوع غير متوقّع في وقت ما، مثل وجود بيانات مخزنة على قرص أو قسم بنظام ملفات NTFS المستخدم في وندوز NT/XP، فلن نكون قادرين على الوصول إليها: لن تعرف النواة كيف تفعل ذلك، أو لن تكون داعمة لهذا العمل. إذا كنّا قد توقعنا ذلك في وقت ما (لكن ليس في العادة) فقد نكون بحاجة للوصول إلى هذه الأنظمة، فيمكننا ترك هذه الأنظمة كوحدات.

3. تصريف النواة

سنبدأ التصريف بالأمر make. سيكون علينا أولاً إنشاء الاعتماديات المحتملة بين الأكواد ومن ثم نوع صورة النواة التي نريدها (في هذه الحالة صورة مضغوطة وهي الحالة العادية):

```
make dep
make bzImage
```

عند الانتهاء من هذه العملية، فسيكون لدينا الجزء المضمَّن من النواة، نحن نفتقد الأجزاء التي ضبطناها كوحدات:

```
make modules
```

في هذه الخطوة نكون أتممنا إعداد وتصريف النواة. هذا الجزء يمكن أن يتم بحساب المستخدم العاديّ أو الجذر، لكن من الآن فصاعداً سنحتاج بالتأكيد لصلاحيات المستخدم الجذر، لأننا سننتقل للجزء المتعلق بالتثبيت.

4. التثبيت

سنبدأ بتثبيت الوحدات:

```
make modules_install
```

وتثبيت النواة الجديدة (من المجلد /usr/src/linux-version/ أو من المسار الذي استخدمناه مؤقتاً):

```
cp arch/i386/boot.bzImage /boot/vmlinux-2.4.0
cp system.map /boot/system.map-2.4.0
```

الملف bzImage هو النواة المصرّفة حديثاً، والتي وضعناها في المجلد /boot/. عادة سنجد النواة القديمة في نفس المجلد /boot/ بالاسم vmlinuz وحده أو متبوعاً برقم الإصدار كرابط رمزي للنواة القديمة. عندما نحصل على نواتنا، يفضّل الإبقاء على النواة القديمة في حال حدوث أيّ خطأ أو في حال عمل الجديدة بشكل سيء، وبهذا يمكننا استعادة القديمة. الملف system.map يحوي الرموز المتاحة للنواة وهو ضروريّ لعملية بدء النواة، وهو موضوع أيضاً في نفس المجلد.

في هذه النقطة، نحتاج أيضاً لاعتبار أنه عند بدء النواة قد تحتاج لإنشاء ملفات من نوع initrd، والتي تخدم كصورة مجمّعة لبعض المشغلات الأساسيّة ويُستخدم عند تحميل النظام، إذا كان النظام يحتاج هذه المشغلات قبل تحميل مكونات معيّنة. هذا ضروريّ في بعض الحالات لنتمكن من إقلاع بقية النظام، حيث يجب تحميل بعض المشغلات في المرحلة الأولى؛

مثل بعض متحكّمات الأقراص من أمثال RAID و متحكمات الوسائط، والتي قد تكون ضروريّة، وبهذا يمكن الوصول إلى القرص لإقلاع بقيّة النظام في المرحلة الثانية.

يمكن إنشاء النواة مع أو بدون صورة initrd وهذا يعتمد على احتياجات العتاد أو النظام. وفي بعض الحالات، قد تتطلّب التوزيعة استخدام صورة initrd، وفي حالات أخرى سيعتمد هذا على العتاد المتوفر لدينا. وهي أيضاً مستخدمة بكثرة للتحكم بحجم النواة، وبهذا يمكن تحميل أجزائها الأساسيّة عبر صورة initrd، ومن ثمّ ما يلي ذلك في المرحلة الثانية على شكل وحدات. في حال الحاجة إلى صور initrd، فيمكن إنشاؤها باستخدام الأداة mkinitrd (انظر إلى دليل الاستخدام man أو إلى الدرس التعليمي في نهاية هذه الوحدة)، وتوضع في المجلد /boot/.

5. الخطوة التالية هي إخبار النظام أي الأنوية عليه الإقلاع بها، لكن هذا يعتمد على نظام إقلاع لينكس:

- من الإقلاع في Lilo، سواء من سجل الإقلاع الرئيسي MBR أو من قسم خاص، علينا إضافة السطور التالية إلى ملف الإعداد (في: /etc/lilo.conf):

```
image = /boot/vmlinuz-2.4.0
    label = 2.4.0
```

حيث image هي النواة التي يجب الإقلاع بها، و label هو اسم الخيار الذي سيظهر عند الإقلاع. يمكننا إضافة هذه السطور أو تعديل سطور النواة القديمة. ننصح بإضافة هذه السطور وإبقاء القديمة، تحسُّباً لحدوث مشكلة، وبهذا يمكن استعادة القديمة. قد يكون لدينا في الملف /etc/lilo.conf إعداد إقلاع واحد أو أكثر للينكس أو لأنظمة أخرى مثل وندوز.

يحدّد كل إقلاع بسطر الصورة والمسمّى الذي يظهر في قائمة الإقلاع. لدينا سطر default = label يحدّد السطر الذي يقلع مبدئيًّا. يمكننا أن نضيف أيضاً root = /dev/... إلى السطر السابق لتحديد قسم القرص الذي يحوي نظام الملفات الرئيسيّ (جذر نظام الملفات '/')، آخذين بعين الاعتبار أن للأقراص أجهزة أسماؤها تشبه /dev/hda (أول قرص ide)، و /dev/hdb، أو /dev/sdx لأقراص SCSI (أو ما يحاكيها[13])، ويتم تحديد القسم بعبارة root = /dev/hda2 إذا كان جذر نظام الملفات '/'

13 يقصد بما يحاكي أقراص SCSI الأقراص التي تستخدم نفس طريقة التخاطب، ومنها iSCSI و SATA وأقراص فلاش USB وقارئات رقاقات الذاكرة

موجوداً في القسم الثاني لقرص IDE الأول. يمكننا أيضاً باستخدام " = append " إضافة معامِلات لإقلاع النواة. إذا كان النظام يستخدم initrd سيكون علينا أيضاً أن نحدد الملف والذي سيكون أيضاً موجوداً في boot/initrd-kernelversion/، مع خيار "= initrd". بعد تغيير إعدادات Lilo نحتاج لكتابتها له ليقلع.

```
/sbin/lilo -v
```

إذا كان لدينا مشكلات، فيمكننا استعادة النواة القديمة باختيار خيارها، ومن ثمّ يمكننا – وذلك بمعاودة الوصول إلى lilo.conf – إعادة الإعداد القديم أو دراسة المشكلة وضبط وإعادة تصريف النواة.

- الإقلاع باستخدام Grub. العملية في هذه الحالة سهلة، حيث سنحتاج لإضافة إعداد جديد مكون من النواة الجديدة وإضافته كخيار آخر إلى ملف grub. يلي ذلك الإقلاع بطريقة مشابهة لما في Lilo، لكن مع تذكر أنه من المجدي تعديل الملف (عادة يكون boot/grub/menu.lst/) وإعادة التشغيل. ومن المفضّل أيضاً إبقاء الإعداد القديم للاستعادة في حال حدوث مشاكل.

SD/MMC/MicroSD... كالتي في الكاميرات والهواتف النقالة وما إلى ذلك.

3.2 الهجرة إلى النواة 2.6

في حال الاضطرار لتحديث أنوية من توزيعات قديمة، أو تغيير جيل النواة باستخدام المصدر البرمجي، علينا أخذ بعض النواحي بعين الاعتبار، وذلك بسبب المستجدات التي تأتي مع فرع 2.6 من النواة.

هذه قائمة ببعض النقاط التي علينا أخذها بعين الاعتبار:

1. غيرت بعض وحدات النواة أسماءها، وبعضها لربما اختفت. علينا تفقّد حال الوحدات المتغيرة المحمّلة (فمثلاً، تفحّص /etc/modules و /etc/modules.conf) وعدّلها لتحاكي التغييرات.
2. أضيفت خيارات جديدة إلى الإعداد الابتدائيّ للنواة: مثل make gconfig، وهي واجهة إعداد مبنية على GTK (التي بنيت عليها جنوم). وفي هذه الحالة علينا البحث في مكتبات جنوم كمتطلّب مسبق. الخيار make xconfig تم تغييره ليعمل بمكتبات QT (التي بنيت عليها كدي).
3. أقل رقم إصدار مطلوب للأدوات المختلفة التي نحتاجها لعملية التصريف تمّت زيادته (راجع Documentation/Changes في مصدر النواة). وخاصة أقل رقم إصدار لمصرّف جنو gcc.
4. الحزمة المبدئيّة لأدوات الوحدات تغيرت لتصبح module-init-tools (بدلاً عن modutils المستخدمة في 2.4). هذه الحزمة متطلب مسبق لتصريف الأنوية 2.6، حيث محمل الوحدات مبني على الإصدار الجديد.
5. نظام devfs صار أثرياً وحلّ محله udev، وهو النظام المتحكم ببدء تشغيل hotplug (توصيل) الأجهزة (والتعرف الأولي عليها، وهي بالأصح محاكاة لعملية وصل الأجهزة عند بدء التشغيل عندما يقلع النظام)، وإنشاء مدخلات في المجلد /dev/ بشكل متغير، لتشمل فقط الأجهزة الموجودة في الوقت الحالي.
6. في دبيان، وتحديداً في إصدارات معينة للفرع 2.6 للصور الثنائية للنواة والترويسات والمصدر البرمجي، تغيرت أسماء الحزم من kernel-images/source/headers إلى linux-images/source/headers.
7. في الحالات الحديثة، الأجهزة التي تتبع تقنيات حديثة (مثل SATA) قد انتقلت بالغالب من /dev/hdX إلى

/dev/sdX. في هذه الحالات، علينا تعديل إعدادات etc/fstab/ ومحمل الإقلاع (lilo أو grub) ليعكس التغييرات.

8. قد تكون هناك بعض المشكلات مع أجهزة إدخال/إخراج معينة. التغيير في أسماء وحدات النواة قد أثر على أجهزة الفأرة – وغيرها – ما قد يؤثر كذلك على X-Window الذي يعمل، إلى أن يتم التحقق من الوحدات المطلوبة وتحميل الوحدات الصحيحة (مثل psmouse). في وقت ما، تضمّنت النواة مشغلات الصوت Alsa. إذا كان لدينا OSS القديم، فسيكون علينا التخلص منها من بين الوحدات التي تحمَّل، حيث تقوم Alsa بمحاكاتها.

9. فيما يتعلق بالمعماريات التي تدعمها النواة، نحتاج لأن نُبقي في بالنا أن النواة 2.6 – بمراجعاتها المختلفة – تعمل على زيادة المعماريات المدعومة، مما يسمح لنا بالحصول على الصور الثنائية للنواة في التوزيعة (أو خيارات تصريف النواة) الأنسب لدعم معالجنا. وبالتحديد، يمكننا إيجاد المعماريات مثل i386 (لإنتل و AMD): وتدعم التوافقية لمعالجات إنتل 32بت لعائلة كاملة من المعالجات (بعض التوزيعات تستخدم 486 كمعمارية أساسيّة)، بعض التوزيعات تتضمن إصدارات مميزة من i686 (إنتل pentium pro والأحدث)، وهناك دعم لـ k7 (لمعالجات AMD Athelon وما يليها)، وهناك دعم لمعالجات 64بت مثل AMD 64 bit، ولمعالجات إنتل مع امتدادات em64t لدعم معالجات 64بت مثل Xeon والمعالجات متعددة الأنوية. في نفس الوقت، هناك معماريات IA64 لمعالجات 64بت نماذج إنتل إيتانيوم Intel Itanium. دعم SMP للمعماريات مفعّل في صورة النواة في معظم الحالات (مالم تكن التوزيعة تدعم إصدارات مع وبدون SMP مُنشأة بشكل مستقل، فإن اللاحقة smp- عادة ما تضاف إلى الصورة التي تدعمها).

10. لإنشاء صور initrd في دبيان، كما في إصدارات معينة للنواة (بدءاً من 2.6.12)، تعتبر أدوات mkinitrd أثرية، وقد حلّت محلها أدوات جديدة، مثل أدوات initramfs و yaird. تسمح كلاهما ببناء صور initrd، لكن الأولى هي التي يُنصح بها (في دبيان).

3.3 تصريف النواة للإصدارات 2.6

في الإصدار 2.6، آخذين بعين الاعتبار ما ذُكِر أعلاه، فإن التصريف يتبع طريقة مشابهة للطريقة المذكورة سابقاً:

بعد تنزيل إصدار للنواة من الفرع 2.6 إلى المجلد الذي سنستخدمه للتصريف وفحص الإصدارات المطلوبة للادوات الأساسيّة، يمكننا التقدّم إلى خطوة تصريف وتنظيف التصريفات السابقة:

```
# make clean mrproper
```

ضبط المعامِلات (تذكر أنه إذا كان لدينا ملف config. سابق، فلن نكون قادرين على بدء الإعداد من الصفر). نقوم بالإعداد من خلال الخيار make (اعتماداً على الواجهة التي نستخدمها):

```
# make menuconfig
```

إنشاء الصورة الثنائية للنواة:

```
# make dep
# make bzImage
```

إنشاء الوحدات (المحددة ليتم تصريفها):

```
# make modules
```

تثبيت الوحدات المُنشأة (/lib/modules/version/)

```
# make modules_install
```

نسخ الصورة إلى مكانها النهائيّ:

```
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.x.img
```

وفي النهاية، إنشاء صورة initrd التي تعتبر ضروريّة، مع الأدوات الضروريّة اعتماداً على الإصدار (انظر إلى التعليقات اللاحقة). وضبط محمل إقلاع Lilo أو Grub، أيهما نستخدم.

الخطوات النهائية (vmlinuz و system.map و initrd) بنقل الملفات إلى /boot/ يمكن بالعادة عملها أيضاً بالعملية:

```
# make install
```

لكن علينا أن نأخذ بعين الاعتبار أن هذا الأمر يقوم بالعملية بأكملها ويحدّث محمّلات الإقلاع، بإزالة أو تعديل الإعداد القديم؛ وفي نفس الوقت، يمكن أن يغير الروابط المبدئيّة في المجلد /boot/. علينا أن نبقي هذا في بالنا عندما يتعلق الأمر بوجود إعدادات سابقة نرغب بالاحتفاظ بها.

فيما يتعلق بإنشاء initrd، يتم إنشاؤها في فيدورا/ردهات آليًّا بالخيار install. أما في دبيان، فعلينا استخدام التقنيات التي في الفصل التالي، أو إنشاؤها باستخدام mkinitrd (للإصدارات 2.6.12 والأقدم)، أو باستخدام mkinitramfs أو أداة تعرف بالاسم update-initramfs، بتحديد إصدار النواة (على اعتبار أنها تُسمى vmlinuz-version في المجلد /boot/):

```
# update-initramfs -c -k 'version'
```

3.4 تصريف النواة في دبيان (طريقة دبيان)

في دبيان، إضافة إلى الطرق المذكورة، سنحتاج لإضافة الإعدادات بطريقة تدعى "طريقة دبيان". وهي طريقة تسمح لنا ببناء النواة بسرعة ومرونة.

نحتاج لبعض الأدوات للقيام بالعملية (بتثبيت الحزم أو ما شابه):

kernel-package, ncurses-dev, fakeroot, wget, bzip2

يمكننا أن نرى العملية من وجهتي نظر: بناء نواة مكافئة للتي توفرها التوزيعة، أو تفصيلها ومن ثمّ استخدام الطريقة لبناء نواة مخصصة مكافئة.

في الحالة الاولى، نبدأ بالحصول على إصدار مصادر النواة الذي توفره التوزيعة (أي المراجعة x للنواة 2.6):

```
# apt-get install linux-source-2.6.x
$ tar -xvjf /usr/src/linux-source-2.6.x.tar.bz2
```

حيث نحصل على المصادر ونفكّ ضغطها (تترك الحزمة الملف في /usr/src/).

تثبيت الأدوات الأساسيّة:

```
# apt-get install build-essentials fakeroot
```

فحص اعتماديات المصدر:

```
# apt-get build-dep linux-source-2.6.x
```

وإنشاء الملفات الثنائيّة، بناء على إعدادات الحزمة المعدّة مسبقاً (مشابهة لتلك المضمّنة في حزم الصورة الرسميّة للنواة في دبيان):

```
$ cd linux-source-2.6.x
$ fakeroot debian/rules binary
```

هناك بعض الإجراءات الإضافيّة لإنشاء الأنوية بناء على المستويات المختلفة للتراقيع التي تقدمها التوزيعة، وإمكانية إنشاء

إعدادات نهائيّة مختلفة (انظر إلى ملاحظة المرجع لإكمال هذه النواحي).

أما في الحالة الثانية، وهي أكثر شيوعاً، عندما نرغب بتخصيص النواة، سيكون علينا أن نتبع عملية مشابهةعبرة خطوات تفصيل اعتياديّة (مثل استخدام make menuconfig)، الخطوات ستكون كالتالي:

الحصول على المجلد وتجهيزه (نحصل هنا على حزمة التوزيعة، لكن يمكن بدلاً من ذلك الحصول على المصدر من kernel.org):

```
# apt-get install linux-source-2.6.x
$ tar xjf /usr/src/linux-source-2.6.x.tar.bz2
$ cd linux-source-2.6.x
```

ومن ثمّ نضبط المعاملات، وكالعادة، يمكننا الاعتماد على ملفات config. التي استخدمناها سابقاً، للبدء من إعداد معروف (للتفصيل، يمكننا أيضاً استخدام أيّ من الطرق الأخرى، مثل xconfig, gconfig, ...)

```
$ make menuconfig
```

الإعداد النهائيّ للنواة سواء بالاعتماد على initrd أو لاً، دون وجود initrd مسبقاً (علينا الانتباه إلى الإصدار الذي نستخدمه، حيث أنه في إصدار معين للنواة، فإن استخدام صورة initrd يمكن ان يكون إلزامياًّ):

```
$ make-kpkg clean
$ fakeroot make-kpkg –revision=custom.1.0 kernel_image
```

أو إذا كان لدينا initrd متاح (مبني مسبقاً)

```
$ make-kpkg clean
$ fakeroot make-kpkg –initrd –revision=custom.1.0 kernel_image
```

ستنتهي العملية بإضافة الحزمة المرتبطة بصورة النواة، والتي سنكون قادرين على تثبيتها في النهاية:

```
# dpkg -i ../linux-image-2.6.x_custom.1.0_i386.deb
```

سنضيف في هذا الفصل أيضاً خاصية أخرى لتؤخذ بعين الاعتبار في دبيان، وهي وجود أدوات من أطراف خارجية لإضافة وحدات النواة المتغيرة. وبالتحديد، تساعد أداة module-assistant على أتمتة هذه العملية بناء على مصادر الوحدات.

سنحتاج لأن تكون ترويسات النواة مثبتة لدينا (الحزمة linux-headers-version) أو المصادر التي نستخدمها

لتصريف النواة. ويمكن كذلك استخدام الأداة module-assistant تفاعليًّا، مما يسمح لنا بالاختيار من قائمة مطوّلة بالوحدات المسجلة مسبقاً في التطبيق، ويمكن أن تكون مسؤولة عن تنزيل الوحدة، وتصريفها، وتثبيتها في النواة الموجودة.

ويمكننا أيضاً أن نحدد من سطر الأوامر ببساطة (حيث m-a مكافئ لـ module-assistant):

```
# m-a prepare
# m-a auto-install module_name
```

والتي تعدّ النظام للاعتماديات المحتملة، وتنزّل مصادر الوحدة، وتصرّفها، وإذا لم تكن هناك مشاكل، تثبيتها للنواة الحاليّة.

يمكننا أن نرى اسم الوحدة في القائمة التفاعليّة لأداة module-assistant.

4 ترقيع النواة

في بعض الحالات يكون تطبيق تراقيع على النواة شائعاً أيضاً.

ملف الترقيع patch ذي العلاقة بنواة لينكس هو ملف آسكي ASCII نصّيّ، يحوي الاختلافات بين المصدر البرمجي الأصلي والأكواد الجديدة، مع معلومات إضافيّة عن عن أسماء الملفات وسطور الكود. برنامج الترقيع (انظر إلى man patch) يساعد في تطبيق هذه التراقيع على شجرة المصدر البرمجي للينكس (والتي عادة ما تكون في /usr/src/).

عادة ما تكون التراقيع ضروريّة عندما يتطلّب عتاد خاص بعض التعديلات على النواة، أو عند اكتشاف علل (أخطاء) إلى إصدار منتشر بكثرة للنواة، أو عند وجود ميزة جديدة معينة نرغب بإضافتها. لتصحيح المشكلة (أو إضافة ميزة جديدة)، من الشائع نشر ترقيع بدلاً من نواة جديدة بأكملها. عندما يكون هناك العديد من هذه التراقيع، يتم إضافتها إلى تحسينات عديدة للنواة السابقة لإنشاء إصدار جديد للنواة. في كلّ الأحوال، إذا كان لدينا عتاد كثير المشاكل، أو إذا كان الخطأ يؤثر على عمل أو استقرار النظام، ولم يكن بإمكاننا الانتظار حتى الإصدار القادم للنواة، فسيكون علينا أن نطبّق الترقيع.

عادة ما يتمّ توزيع الترقيع في ملف مضغوط من نوع bz2 (استخدم bunzip2 لفكّه، لكن يمكنك أيضاً أن تجده بامتداد gzip وهو gz.)، في هذه الحالة على سبيل المثال:

patchxxxx-2.6.21-pversion.bz2

حيث يكون xxxx رسالة ما تتعلّق بنوع أو هدف الترقيع، و 2.6.21 هو إصدار النواة الذي يطبَّق عليه الترقيع، و pversion تشير إلى رقم إصدار الترقيع، والذي يمكن أن يحمل عدّة إصدارات. علينا أن نبقي في بالنا أننا نتحدث عن تطبيق التراقيع على مصادر النواة (والتي عادة ما تكون مثبتة – كما رأينا سابقاً – في /usr/src/linux/ أو مجلد مشابه).

ما إن نحصل على الترقيع فسيكون علينا تطبيقه، وسنجد العملية التي علينا اتباعها في ملف readme الذي يرافق الترقيع، لكن عادة ما تتبع العملية (بعد التأكد من توفر المتطلبات المسبقة لها) خطوات فكّ ضغط الترقيع في مجلد الملفات المصدرية، وتطبيقه على مصادر النواة، على سبيل المثال:

```
cd /usr/src/linux        (أو /usr/src/linux-2.6.21 أو أيّ إصدار).
bunzip2 patch-xxxxx-2.6.21-version.bz2
patch -p1 < patch-xxxxx-2.6.21-version
```

وبعد ذلك سيكون علينا أن نعيد تصريف النواة لإنشائها مرّة أخرى.

يمكن الحصول على التراقيع من أماكن مختلفة. عادة يمكننا إيجادها في موقع التخزين للنواة (http://www.kernel.org)، أو في http://www.linuxhq.com، والذي فيه سجل كامل بها. وتوفّر بعض مجتمعات لينكس (ومستخدمون أفراد) تصحيحات أيضاً، لكن من الأفضل البحث في المواقع القياسيّة وذلك للتأكد من أن التراقيع موثوقة ولتجنب المشاكل الأمنية المحتملة من تراقيع "القراصنة". ومن الطرق الأخرى مصنّع العتاد، والذي قد يقدّم تعديلات معيّنة على النواة (أو المتحكّمات) وذلك لكي تعمل أجهزته بشكل أفضل (ومن الأمثلة المعروفة على ذلك إنفيديا NVIDIA ومشغّلات الأجهزة لبطاقات الرسوميّات التابعة لها).

وفي النهاية، علينا أن نذكر بأن كثيراً من توزيعات جنو/لينكس (فيدورا/ردهات، ماندريفا، ...) توفّر أنوية مرقّعة من طرفهم وأنظمة لتحديثها (وبعض هذه الأنظمة تحدّث الأنوية آليًّا، كما هي الحال في فيدورا/ردهات ودبيان). عادة ما ينصح في الأنظمة الإنتاجيّة بمتابعة تحديثات المصنّع[14]، رغم أنه لا يوفر بالضرورة أحدث نواة منشورة، بل النواة التي يجدها أكثر استقراراً مع توزيعته، على حساب فقدان مزايا أحدث الاجيال أو التقنيات المستحدثة المضمّنة في النواة.

14 يقصد بالمصنّع هنا صائني التوزيعة، وليس مصنّعي العتاد.

5 وحدات النواة

للنواة القدرة على تحميل أجزاء متغيّرة من الكود (وحدات) عند الحاجة، وذلك لإكمال وظائفها (هذه الإمكانيّة متوفّرة منذ الإصدار 1.2 من النواة). فمثلاً، يمكن أن تضيف الوحدة دعم نظام ملفات أو جهازاً (عتاداً) معيّناً. عندما لا تكون الميزة التي توفّرها الوحدة ضروريّة، فيمكن تنزيلها لتحرير الذاكرة.

عند الطلب، تحدد النواة الخاصية غير المتوفرة في النواة في تلك اللحظة، وتقوم بالاتصال بأحد خيوط النواة ويعرف بـ kmod (في إصدارات الفرع 2.0 من النواة كان يطلق على المراقب kerneld)، والذي ينفّذ الأمر modprobe في محاولة لتحميل الوحدة من السلسلة باسم الوحدة أو عبر معرّف عام، هذه المعلومة موجودة في الملف /etc/modules.conf على شكل اسم مستعار بين الاسم والمعرّف.

ومن ثمّ نبحث في /lib/modules/version_kernel/modules.dep لنرى إذا كان هناك اعتماديّات من وحدات أخرى. وفي النهاية، يتم تحميل الوحدة من المسار /lib/modules/version_kernel/ (المجلد القياسيّ للوحدات) بالأمر insmod، حيث version_kernel هو إصدار النواة الحالية والذي يتم ضبطه عبر الأمر uname -r. وبهذا، فالوحدات بهيئتها الثنائيّة مرتبطة بإصدار معيّن للنواة، وهي عادة موجودة في /lib/modules/version_kernel/.

إذا كنا بحاجة لتصريفها، فسنحتاج للحصول على المصادر و/أو ترويسات الإصدار المصمّمة لها.

هناك بعض الأدوات التي تسمح لنا بالتعامل مع الواحدات (عادة ما تظهر في حزمة برمجيات اسمها modutils، والتي حلّت محلها module-init-tools لإدارة وحدات الفرع 2.6):

1. **lsmod** : يمكنا أن نرى الوحدات المحمّلة في النواة (يتم الحصول على المعلومات من الملف الوهمي /proc/modules). وتظهر الاسم، والاعتماديات (داخل أقواس مربّعة [ ])، وأحجام الوحدات بالبايت، وعدّاد استخدام الوحدات، هذا يسمح لها بأن تحمَّل إذا كان العدّاد صفراً.

مثال

بعض الوحدات في توزيعة دبيانيّة:

Module	Size	Used by	Tainted: P
agpgart	37.344	3	(autoclean)
apm	10.024	1	(autoclean)
parport_pc	23.304	1	(autoclean)
lp	6.816	0	(autoclean)
parport	25.992	1	[parport_pc lp]
snd	30.884	0	
af_packet	13.448	1	(autoclean)
NVIDIA	1.539.872	10	
es1371	27.116	1	
soundcore	3.972	4	[snd es1371]
ac97_codec	10.9640	0	[es1371]
gameport	1.676	0	[es1371]
3c59x	26.960	1	

2. **modprobe** : يحاول تحميل وحدة واعتمادياتها.

3. **insmod** : يحمّل وحدة معيّنة.

4. **depmod** : يحلّل الاعتماديات بين الوحدات وينشئ ملفاً بالاعتماديات.

5. **rmmod** : يزيل وحدة من النواة.

6. أوامر أخرى يمكن استخدامها لتصحيح وتحليل الوحدات، مثل modinfo، والتي تعرض قائمة ببعض المعلومات ذات العلاقة بالوحدة، أو ksyms والتي (فقط في إصدارات 2.4) تسمح لنا باختبار الرموز المصدّرة من الوحدات (وهي موجودة أيضاً في /proc/ksyms).

عادة ما يتم تحديد اسم الوحدة لتحميلها، سواء عبر النواة نفسها، أو بتنفيذ المستخدم للأمر insmod ومعاملات اختياريّة محدّدة. على سبيل المثال، في حالة الأجهزة، فمن الشائع تحديد عناوين منافذ الدخل والخرج أو مصادر DMA أو IRQ. فمثلاً:

```
insmod soundx io = 0x320 irq = 5
```

6 مستقبل النواة والبدائل

في أوقات معينة، كان يتم إصدار تطويرات لينكس على فترات زمنية قصيرة جدًّا، لكن الآن مع الوضع المستقر لسلسلة 2.6 من النواة، فقد ازداد الوقت المنقضي بين إصدارات النواة، والذي يعد - بطريقة ما - إيجابيًّا. هذا يسمح بوجود وقت لتصحيح الأخطاء، ورؤية أيّ الأفكار لم تعمل جيداً، وتجربة أفكار جديدة يتم تضمينها إذا عملت.

سنناقش في هذا الفصل بعض أفكار الأنوية الأخيرة، وبعض تلك المخطط لها في المستقبل القريب في تطوير النواة.

السلسلة السابقة (سلسلة 2.4) التي كان يتم تضمينها في التوزيعات قبل سنوات[15]، كانت تتم المساهمات في:

1. إكمال مقاييس معيار IEEE POSIX، وهذا يعني أن العديد من برامج يونكس الموجودة مسبقاً يمكن إعادة تصريفها وتشغيلها على لينكس.
2. تحسين دعم الأجهزة: التوصيل والتشغيل[16]، والمنفذ التسلسلي العالمي USB، والمنفذ المتوازي[17]، SCSI، ...
3. دعم أنظمة الملفات الحديثة، مثل UDF (الأقراص الضوئيّة القابلة لإعادة الكتابة كما في الأقراص الصلبة)، وأنظمة [ملفات] سجلّية أخرى، مثل Reiser من IBM أو ext3، ويسمح هذا بالحصول على تقرير (سجلّ) بتغييرات نظام الملفات، وبهذا يمكن الاسترجاع عند وجود أخطاء أو تعامل غير سليم مع الملفات.
4. دعم الذاكرة حتى 4 جيجا، حيث ظهرت مشاكل في وقت ما (مع أنوية 1.2) بحيث لم تتمكن النواة من دعم أكثر من 128 ميجا (كان يعتبر هذا القدر من الذاكرة أيامها كبيراً جدًّا).
5. تحسنت واجهة /proc/. مجلد /proc/ نظام ملفات وهمي ليس موجوداً على القرص فعليًّا، ولكنه ببساطة للوصول إلى بيانات النواة والعتاد بشكل منظّم.
6. دعم الصوت في النواة: متحكمات Alsa - التي كانت تضبط بشكل مستقل سابقاً - تم إضافتها جزئيًّا.

---

15 هنا تعديل على النصّ الأصليّ.

16 التوصيل والتشغيل: Plug and Play ويشار إليها اختصاراً PnP.

17 المنفذ المتوازي Parallel Port، هو المنفذ العريض الذي كان يستخدم لوصل الطابعات في التسعينات.

7. تم تضمين دعم تمهيديّ لبرمجيات RAID ومدير الوسائط المتغيرة LVM1.

في السلسلة الحالية، قدّم الفرع 2.6 من النواة تحسينات هامّة مقارنة بسابقه (مع المراجعات المختلفة للفرع 2.6):

1. مزايا SMP محسّنة، وهي هامّة للمعالجات متعدّدة النوى المستخدمة بكثرة في البيئات العلميّة والأعمال [وحتى الاستخدام الشخصيّ].
2. تحسينات في الجزء المتعلّق بجدولة المعالج.
3. تحسينات في دعم تعدّد الخيوط لتطبيقات المستخدم. استُخدِمَت نماذج جديدة للخيوط مثل NGPT من IBM و NPTL من ردهات (ومع الوقت تم الاستقرار في النهاية على NPTL).
4. دعم USB 2.0 [و USB 3.0][18].
5. اعتماد متحكمات صوت Alsa في النواة.
6. معماريات جديدة لمعالجات 64بت، حيث دُعمت معالجات AMD x86_64 (والتي تعرف أيضاً بمسمّى amd64) و PowerPC 64 و IA64 (معمارية إنتل إيتانيوم).
7. دعم أنظمة الملفات السجلية: JFS، و JFS2 (من IBM)، و XFS (من Silicon Graphics).
8. تحسين مزايا الدخل/الخرج، ونماذج جديدة لمتحكمات موحدة.
9. تحسينات في تنفيذ TCP/IP، ونظام NFSv4 (مشاركة نظام الملفات مع أنظمة أخرى عبر الشبكة).
10. تحسينات واضحة للنواة المتساهلة: هذا يسمح للنواة بإدارة العديد من المهام داخلياً بحيث تتقاطع مع بعضها، وهذا ضروريّ للتنفيذ الفعال لأنظمة الوقت الحقيقيّ.
11. تعليق النظام واسترجاعه عند إعادة التشغيل (من النواة).

---

18 في الحقيقة، لقد كان لينكس هو أول نظام يدعم USB 3.0 وذلك في شهر أيلول/سبتمبر عام 2009، وأول نظام يدعم أول قرص صلب يعمل بهذه التقنية والذي عرض في معرض في اليابان في الشهر نفسه.

12. لينكس في وضع المستخدم User Mode Linux – UML، وهو نوع من أجهزة لينكس الوهمية على لينكس، والتي تسمح لنا برؤية لينكس (في وضع المستخدم) يعمل على جهاز وهميّ. هذا مثاليّ للتصحيح، حيث يمكن بذلك تطوير واختبار إصدار لينكس على نظام آخر، وهذا مفيد لتطوير النواة نفسها ولتحليل أمنها.
13. تقنيات المحاكاة مضمّنة في النواة: لقد كانت التوزيعات تقوم تدريجيًّا بتضمين تقنيات مختلفة للمحاكاة، والتي تتطلب امتداداً في النواة؛ علينا أن نذكر – على سبيل المثال – الأنوية المعدّلة لكل من Xen، أو Virtual Server المعروف أيضاً بالاسم Vserver[19].
14. إصدار جديد من دعم الوسائط LVM2.
15. نظام ملفات وهمي جديد /sys/، مصمم لتضمين معلومات النظام والأجهزة التي سيتم ترحيلها من نظام /proc/، مما يترك هذا الأخير بمعلومات عن العمليات وتطورها أثناء التنفيذ.
16. وحدة FUSE لتنفيذ أنظمة ملفات في مساحة المستخدم (وعلى رأسها حالة NTFS)[20].

في المستقبل، هناك خطة لعمل تحسينات في النواحي التالية[21]:

1. زيادة تقنيات المحاكاة في النواة، لدعم إعدادات أنظمة تشغيل مختلفة، وتقنيات محاكاة مختلفة، إضافة إلى دعم أفضل للعتاد للمحاكاة المضمّنة في المعالجات التي تظهر في المعماريات الجديدة.
2. دعم SMP (الأجهزة متعدّدة المعالجات) لمعالجات 64بت (إنتل إيتانيوم Intel Itanium، وأوبتيرون AMD Opteron)، ودعم المعالجات المتعدّدة.
3. أنظمة ملفات محسّنة للشبكات العنقوديّة والأنظمة الموزّعة.

---

19 وهناك تقنيات أخرى للمحاكاة في لينكس منتشرة هذه الأيام أيضاً، ومنها OpenVZ و Xen و KVM وغيرها ...

20 يشير FUSE اختصاراً إلى Filesystem in User Space، وله استخدامات أخرى غير NTFS، كاختبار أنظمة ملفات جديدة قيد التطوير دون الاضطرار لتضمينها في النواة، وللوصول إلى ملفات نظام بعيد ووصلها كنظام ملفات محلّي عبر بروتوكول ssh فيما يعرف بنظام ملفات sshfs، وهناك أيضاً نظام ملفات سامبا smb/cifs، والذين يمكن ضمهما إلى شجرة الملفات كما لو كانا نظام ملفات محليّ.

21 تذكر أن الكتّاب قديم، وأن بعض أو كلّ هذه النواحي قد تكون تمّت بالفعل.

4. تحسينات للأنظمة الموجّهة للأجهزة النقّالة (كالمساعدات الشخصيّة PDA، والهواتف النقّالة، ...).
5. تحسين دعم معيار POSIX.
6. تحسينات في جدولة المعالج، رغم أن العديد من التحسينات تم عملها منذ بداية الفرع 2.6، إلا أنه ما زال هناك أداء ضعيف في بعض الحالات، وعلى وجه التحديد في استخدام تطبيقات واجهة المكتب، وهناك العديد من البدائل التي يجري دراستها لتحسين هذه الناحية ونواحي أخرى.

إضافة إلى هذا، وبعيداً عن أنظمة لينكس، فكلّ من مؤسسة البرمجيات الحرّة FSF ومشروعها جنو GNU مازالا يعملان على المشروع للخروج بنظام تشغيل كامل. من المهم تذكّر أن الحصول على بديل برمجيّ حرّ لنظام يونكس كان من الأهداف الأساسيّة لمشروع جنو، وأن أدوات جنو ما هي إلا برمجيات ضروريّة للنظام. عام 1991، عندما قام لينكس بدمج نواته ببعض أدوات جنو، اتُّخِذَت الخطوة الأولى نحو أوج ما وصلت إليه أنظمة جنو/لينكس. لكن مشروع جنو ما يزال يعمل على فكرته للخروج بنظام كامل. والآن لديهم نواة يمكنها تشغيل أدوات جنو. وتعرف هذه النواة باسم هيرد Hurd، والنظام المبني بها يسمى جنو/هيرد. هناك بالفعل بعض التوزيعات التجريبيّة، وتحديداً نظام دبيان جنو/هيرد.

لقد تم تصميم هيرد كنواة لنظام جنو حوالي عام 1990، حيث بدأ تطويره آنذاك، وحيث كانت معظم برمجيات جنو قد تمّ تطويرها في ذلك الوقت، والشيء الوحيد الذي كان ناقصاً هو النواة. [وبعد ذلك بعام، أي] عام 1991، دمج لينُس جنو بنواة لينكس، مما نتج عنه بداية تاريخ أنظمة جنو/لينكس. لكن استمر هيرد بالتطور. أفكار تطوير هيرد معقّدة أكثر، حيث يمكن اعتبار لينكس ذا تصميم محافِظ، مبني على أفكار معروفة ومنفّذة مسبقاً.

وعلى وجه التحديد، وضع تخيُّل هيرد كمجموعة من الخوادم المنفَّذة على النواة المصغّرة ماتش Mach، وهي نواة مصمّمة على نمط الأنوية المجزَّأة (عكس لينكس ذي النواة الأحاديّة) الذي طورته جامعة كارنيجي مِلُن University of Carnegie Mellon، والذي تلتها فيه جامعة Utah. كانت الفكرة الأساسيّة هي جعل وظائف نواة يونكس على شكل خوادم يمكن تنفيذها على نواة أساسيّة بسيطة تُدعى Mach. تأجل تطوير هيرد ريثما يتم الانتهاء من تصميم ماتش، والتي تم نشرها في النهاية كبرمجيّة حرّة مما

سمح باستخدامها في تطوير هيرد. في ذلك الوقت، علينا أن نذكر أهمّيّة ماتش، حيث أن العديد من أنظمة التشغيل مبنية الآن على أفكار تم استقاؤها منه، وأكثر هذه الأمثلة قوّة نظام Apple MacOS X.

تأخر تطوير هيرد أكثر بسبب التعقيدات الداخلية، ولأنه كان لها عدّة خوادم بمهام مختلفة من النوع متعدّد الخيوط (تنفيذ عدّة خيوط)، وكان التصحيح صعباً للغاية. لكن الآن، الإصدارات الإنتاجيّة الأولى لجنو/هيرد متاحة بالفعل، إضافة إلى إصدارات اختباريّة لتوزيعة جنو/هيرد.

قد تتواجد في مستقبل ليس ببعيد أنظمة جنو/لينكس بشكل مشترك مع جنو/هيرد، أو أن نواة لينكس ستستبدل لتحل محلها نواة هيرد، إذا نجحت بعض القضايا ضدّ لينكس (اقرأ قضية SCO ضد IBM)، حيث ستقدّم حلًّا لتجنب المشكلات المستقبليّة. في جميع الأحوال، لكلي النظامين مستقبل واعد أمامهما. سنعلم مع الوقت كيف ستسير الأمور.

7 درس تعليمي: إعداد النواة حسب متطلبات المستخدم

سنلقي في هذا الفصل نظرة على ورشة عمل تفاعليّة صغيرة لعملية تحديث وضبط النواة في التوزيعتين المستخدمتين: دبيان وفيدورا.

الشيء الأول الضروريّ – قبل البدء – هو معرفة الإصدار الحاليّ للنواة التي لدينا بالأمر uname -r، لنعلم ما الإصدار التالي الذي نرغب بالتحديث إليه أو تخصيصه. والشيء الآخر هو توفر طريقة لإقلاع نظامنا في حال وجود أخطاء: كوجود أقراص التثبيت الضوئيّة، أو قرص الصيانة (حالياً يستخدم القرص الضوئيّ الأول للتوزيعة لهذا الغرض) أو قرصاً ضوئيًّا حيًّا لتوزيعة ما يسمح لنا بالوصول إلى نظام ملفات الجهاز، وذلك لإعادة عمل أيّة إعدادات يمكن أن تكون قد سبّبت لنا مشكلة. من المهم أيضاً أخذ نسخة احتياطيّة لبياناتنا أو الإعدادات الهامّة.

سننظر في الاحتمالات التالية:

1. تحديث نواة التوزيعة. الحالة الآليّة في دبيان.
2. التحديث الآليّ في فيدورا.
3. تطويع نواة عامّة (دبيان أو فيدورا). في الحالة الأخيرة هذه، الخطوات بالأساس نفسها كتلك المعروضة في فصل الإعداد، لكننا سنقوم بإبداء بعض التعليقات الإضافيّة.

7.1 ضبط النواة في دبيان

في حالة توزيعة دبيان، يمكن للتثبيت أن يكون آليًّا أيضاً، وذلك باستخدام نظام الحزم APT. يمكن عمل ذلك عبر سطر الأوامر، أو عبر مدراء APT رسوميين (مثل synaptic أو gnome-apt [أو PackageKit] ...).

سنشرح عملية التثبيت عبر سطر الأوامر باستخدام apt-get، على فرض أن الوصول إلى مصادر apt (وعلى رأسها مصادر دبيان الأصليّة) مضبوط بشكل صحيح في ملف /etc/apt/sources.list. لننظر إلى الخطوات:

1. لتحديث قائمة الحزم.

```
# apt-get update
```

2. قائمة الحزم المرتبطة بصور النواة:

```
# apt-cache search linux-image
```

3. اختيار إصدار مناسب لمعماريّتنا (عامّ، أو 386/486/686 من إنتل، أو k6 أو k7 من AMD، أو على وجه التحديد إصدارات 64بت amd64 من إنتل AMD، و ia64 لمعالجات إنتل إيتانيوم). يتكون الإصدار من إصدار النواة، ومراجعة دبيان للنواة، والمعمارية. مثال: kernel-2.6.21-4-k7، وهي نواة لمعالج AMD Athelon، مراجعة دبيان الرابعة للإصدار 2.6.21 من النواة.

4. تفحّص توفر الوحدات الإضافية الملحقة للإصدار المحدّد (بنفس رقم الإصدار). سنبحث باستخدام apt-cache عما إذا كانت هناك وحدات متغيرة أخرى يمكن أن تكون مفيدة لعتادنا، اعتماداً على إصدار النواة التي نريد تثبيتها. تذكر أنه كما رأينا في طريقة دبيان، هناك أيضاً أداة module-assistant، والتي تسمح لنا بأتمتة هذه العملية بعد تصريف النواة. إذا لم تكن الوحدات الضروريّة مدعومة، فقد يمنعنا هذا من تحديث النواة إذا كنّا نعتبر أن عمل عتاد معين تكثر معه المشاكل ضروريًّا للنظام.

5. إذا كنا نرغب أيضاً بالحصول على المصدر البرمجي للنظام، فابحث أيضاً في Linux-source-version (فقط 2.6.21، وهو الرقم الأساسيّ) وترويسات النواة المتوافقة معها، في حال أردنا عمل نواة مخصّصة لاحقاً: في هذه الحالة، النواة العامة ذات العلاقة والمرقّعة من طرف دبيان.

6. ثبّت ما قررنا تثبيته: إذا قررنا التصريف من المصدر، أو ببساطة الحصول على المصدر:

```
# apt-get install linux-image-version
# apt-get install xxxx-modules-version
```
(إذا كانت بعض الوحدات ضروريّة)

و

```
# apt-get install linux-source-version-generic
# apt-get install linux-headers-version
```

7. ثبّت النواة الجديدة، في محمّل الإقلاع Lilo على سبيل المثال (تفحّص أداة الإقلاع المستخدمة، فدبيان تستخدم حاليًّا محمل الإقلاع Grub-legacy، وربما أصبحت تستخدم Grub2 بينما تقرأ هذا الكتاب[22])، وهذا يتمّ آليًّا. إذا سئلنا عمّا إذا كانت initrd فعالة فعلينا التأكد من ملف lilo (وهو /etc/lilo.conf)[23]، وتضمين السطر الجديد في ملف إعداد الصورة الجديدة في lilo، وهو:

```
initrd = /initrd.img-version (or /boot/initrd.img-version)
```

ما إن يتمّ ذلك، فسيكون علينا الحصول على lilo في الوضع fragment، على فرض أن initrd.img و vmlinuz روابط لمكان ملفات النواة الجديدة:

---

22 تم تعديل هذه النقطة. يرجى الانتباه إلى أن معظم التوزيعات الحديثة حاليًّا تستخدم Grub2، وبعض الإصدارات ربما ما تزال تستخدم grub-legacy، أما Lilo فاستخدامه نادر جدًّا هذه الأيّام.

23 بالنسبة لمحمل لإقلاع grub-legacy، فيمكن إيجاد ملف إعداد الإقلاع في /boot/grub/grub.cfg أو في /boot/grub/menu.lst. لضبط grub2، يرجى الاطّلاع على دليل محمّل الإقلاع Grub2، أو على النسخة المترجمة (العربيّة) منه.

```
default = Linux

image = /vmlinuz
    label = Linux
    initrd = /initrd.img
#   restricted
#   alias = 1
image = /vmlinuz.old
label = LinuxOLD
initrd = initrd.img.old
#   restricted
#   alias = 2
```

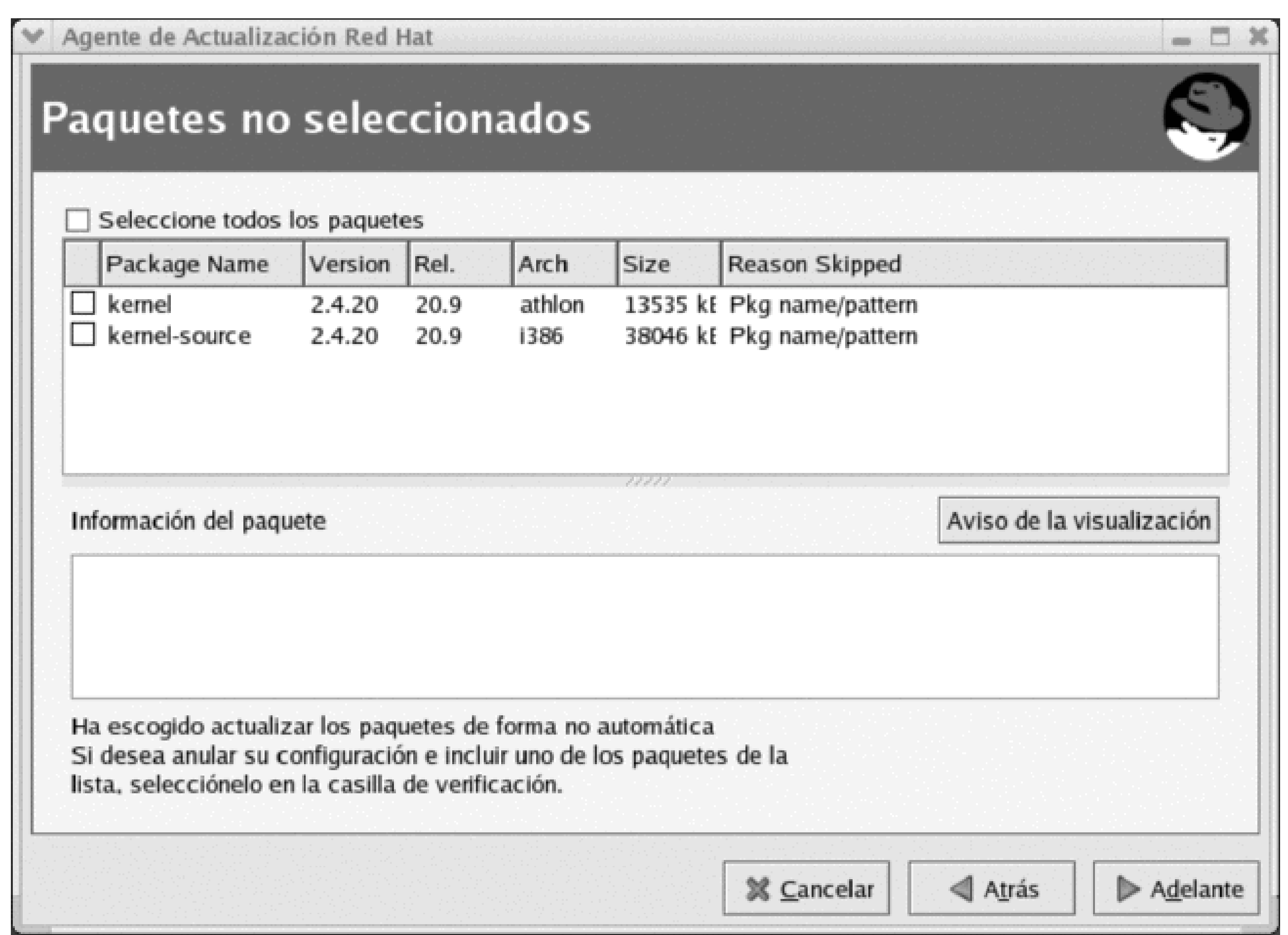
لدينا النواة الأولى مبدئيّة، والثانية هي النواة السابقة. وبهذا، فيمكننا من قائمة lilo طلب أيّ منهما، أو يمكننا ببساطة استعادة القديمة بتغيير المبدئيّة. في أيّ وقت نعدّل فيه على etc/lilo.conf/، فعلينا أن لا ننسى أن نعيد كتابة القسم المتعلّق به باستخدام الأمر sbin/lilo/ أو sbin/lilo -v/.

7.2 ضبط النواة في فيدورا/ردهات

تحديث النواة في توزيعة فيدورا/ردهات آليّ بالكامل عبر خدمة إدارة الحزم فيها، أو عبر البرامج الرسوميّة للتحديث التي تضمّنها التوزيعة؛ على سبيل المثال، في إصدارات الأعمال من ردهات، هناك أداة اسمها up2date. عادة سنجدها في شريط المهام، أو في قائمة أدوات النظام في فيدورا/ردهات (تفقّد الأدوات المتاحة في قوائم أدوات النظام [حسب الواجهة لديك]، الادوات الرسومية المتوفرة حاليًّا تعتمد بشكل كبير على إصدار التوزيعة).

برنامج التحديث هذا يتفقّد بشكل أساسيّ حزم التوزيعة الحاليّة، ويقارنها بقاعدة بيانات فيدورا/ردهات، ويقدّم إمكانيّة تنزيل حزم محدّثة، بما فيها حزم النواة. تعمل هذه الخدمة في إصدار ردهات للأعمال عبر حساب في الخدمة، وتوفرها ردهات بمقابل ماديّ. مع هذا النوع من الأدوات، يتم تحديث النواة آليًّا.

على سبيل المثال، في الشكل 10، نرى أنه بمجرد تشغيل أداة التحديث، فقد تم اكتشاف إصدار جديد متاح للنواة، يمكننا اختياره للتنزيل:



شكل 3: تُظهر خدمة تحديث ردهات (up2date لشبكة ردهات) تحديث النواة المتاح ومصادره

يمكننا في فيدورا إما استخدام الأدوات الرسوميّة المكافئة، أو ببساطة استخدام yum مباشرة، إذا كنا نعلم أن هناك أنوية جديدة متاحة:

```
# yum install kernel kernel-source
```

بمجرد تنزيلها، ننتقل لمرحلة التثبيت، وهي بالعادة آليّة أيضاً، بغض النظر عن نوع محمّل الإقلاع. في حالة grub، عادة ما تكون العمليّة آليّة، وتنتهي بوجود زوجين من الخيارات في القائمة، أحدهما للإصدار الجديد، والآخر للإصدار القديم. على سبيل المثال، في إعداد grub-egacy هذا (الملف في /boot/grub/grub.conf أو في /boot/grub/menu.lst) لدينا نواتان مختلفتان، مع أرقام الإصدارات المتعلّقة بها.

```
# file grub.conf
default = 1
timeout = 10
splashimage = (hd0,1)/boot/grub/splash.xpm.gz

title Linux (2.6.20-2945)
root (hd0,1)
kernel /boot/vmlinuz-2.6.20-2945 ro root = LABEL = /
initrd = /boot/initrd-2.6.20-18.9.img

title LinuxOLD (2.6.20-2933)
root (hd0,1)
kernel /boot/vmlinuz-2.4.20-2933 ro root = LABEL = /
initrd = /boot/initrd-2.4.20-2933.img
```

يتضمن كلّ إعداد عنواناً يظهر عند بدء التشغيل. الجذر أو الجزء من القرص الذي يتم الإقلاع منه، وهو المجلد الذي يحوي ملف النواة وملف initrd.

في حال وجود lilo (حيث grub مستخدم مبدئيًّا) في فيدورا/ردهات كمحمل إقلاع، فسيحدّثه النظام أيضاً (الملف etc/lilo.conf/، لكن بعدها سيكون علينا إعادة كتابة الإقلاع يدويًّا بالامر sbin/lilo/.

من المهم أيضاً أن نذكر أنه كان لدينا في التثبيت السابق إمكانية تنزيل مصادر النواة؛ وهذه - بمجرد تثبيتها - تكون موجودة في usr/src/linux-version/ ويمكن تصريفها وإعدادها باتباع الإجراءات الاعتياديّة كما لو كانت نواة عامّة. علينا أن نذكر أيضاً أن شركة ردهات تقوم بكثير من العمل على ترقيعات وتصحيحات للنواة (المستخدمة في فيدورا) وأن أنويتها تعديلات على المعيار العام مع عدد من الإضافات، مما يعني أنه قد يكون من الأفضل استخدام مصادر ردهات ذاتها، إلا إذا كنّا نريد نواة أحدث أو تجريبيّة أكثر من تلك التي تقدمها ردهات.

7.3 ضبط نواة عامّة

لننظر إلى الحالة العامة لتثبيت نواة بدءاً من مصادرها. لنفترض أن لدينا بعض المصادر مثبتة في /usr/src/ (أو في مكان ما). عادة سيكون لدينا مجلد بالاسم linux مع أو بدون رقم الإصدار، أو رقم الإصدار وحده. ستكون هذه شجرة مصادر النواة.

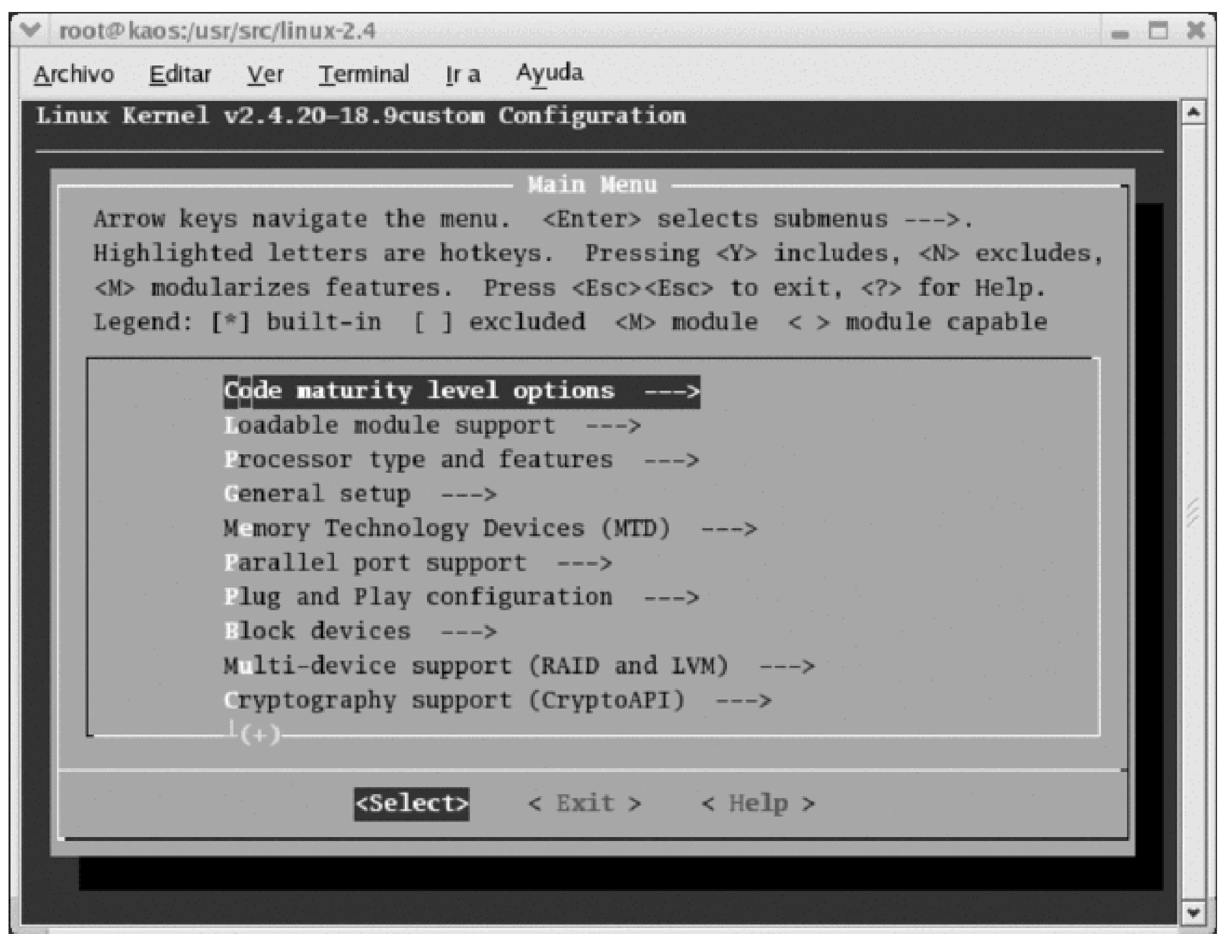
يمكن أن تأتي هذه المصادر من التوزيعة نفسها (أو يمكن أن نكون قد نزّلناها أثناء التحديث السابق). في البداية، سيكون من المهم التأكد إذا كانت أحدث إصدار متوفر، كما فعلنا سابقاً في فيدورا أو دبيان. أو إذا كنا نريد الحصول على أحدث الإصدارات العامّة، فيمكننا الذهاب إلى الموقع www.kernel.org وتنزيل أحدث إصدار متاح (ويفضل أن تكون المستقرة وليس التجريبيّة، ما لم نكن مهتمين بتطوير النواة). ننزل الملف، ونفكّ ضغط مصادر النواة في /usr/src/ (أو الأفضل أن يتم ذلك في مجلد آخر نختاره). يمكننا أيضاً أن نبحث لنرى إذا كانت هناك تراقيع للنواة ونطبّقها (كما رأينا في القسم 4.4 من هذا الجزء).

فيما يلي، سنعلّق على الخطوات التي سنحتاج لتوضيحها: سنعلّق بإيجاز، حيث أن كثيراً منها ذُكِرَت قبلاً عندما كنّا تحدّث عن الإعداد والتفصيل.

1. تنظيف المجلد من الاختبارات السابقة (إذا كان هناك اختبارات سابقة):

```
make clean mrproper
```

2. ضبط النواة، باستخدام الأمر make menuconfig مثلاً (أو xconfig، أو gconfig، أو oldconfig). رأينا هذا في القسم 4.3 من هذا الجزء.



شكل 4: ضبط النواة باستخدام القوائم النصيّة

3. الاعتماديات وتنظيف التصريفات السابقة:

make dep

4. التصريف وعمل صورة للنواة: بالأمر make bzImage. من الممكن استخدام zImage أيضاً إذا كانت الصورة صغيرة، لكن استخدام bzImage شائع أكثر، كما وتحسن عملية التحميل وضغط الأنوية الأكبر. قد لا تعمل bzImage على بعض الأجهزة الأثرية، وبهذا تكون zImage ضروريّة. يمكن أن تستغرق العملية من بضع دقائق وحتى ساعة على الأجهزة الحديثة، وقد تستمر لساعات على الأجهزة الأقدم. وعندما تنتهي، يمكن إيجاد الصورة في:

/usr/src/directory-sources/arch/i386/boot/.

5. يمكننا الآن تصريف الوحدات بالأمر make modules. حتى الآن لم نغير شيئاً في نظامنا. علينا الآن الانتقال إلى خطوة التثبيت.

6. في حالة الوحدات، علينا أن نكون حذرين عندما تجرّب إصداراً قديماً جدًّا من النواة (مثل الفرع 2.2 أو المراجعات الأولى للفرع 2.4)، حيث كانت بعضها تستبدل الأقدم منها (أما في الإصدارات الحديثة - بدأً من بعض مراجعات الإصدار 2.4 - لم تعد هذه المشكلة موجودة).

لكن سيكون علينا أن نكون حذرين أيضاً إذا كنّا سنصرّف إصداراً مطابقاً (أي يحمل نفس رقم الإصدار تماماً) لما لدينا (سيتم استبدال الوحدات)، فمن الأفضل أن نأخذ نسخة احتياطيّة عن الوحدات:

```
cd /lib/modules
tar cvzf old_modules.tgz versionkernel-old/
```

في هذه الحالة صار لدينا نسخة في ملف tgz. يمكننا استعادتها لاحقاً إذا واجهنا أيّة مشكلة، وفي النهاية، يمكننا تثبيت الوحدات بالأمر:

```
make modules install
```

7. يمكننا الآن الانتقال إلى تثبيت النواة، وذلك على سبيل المثال بالأوامر التالية[24]:

```
# cd /usr/src/directory-sources/arch/i386/boot
# cp bzImage /boot/vmlinuz-versionkernel
# cp System.map /boot/System.map-versionkernel
# ln -s /boot/vmlinuz-versionkernel /boot/vmlinuz
# ln -s /boot/System.map-versionkernel /boot/System.map
```

بهذه الطريقة نخزن الملف الرمزيّ للنواة (system.map) وصورة النواة.

8. كل ما علينا فعله الآن هو وضع الإعدادات المطلوبة في ملف إعداد مدير الإقلاع، سواء كان lilo أو grub-legacy أو Grub2، بما يوافق الإعداد الذي رأيناه سابقاً عندما شرحنا عن فيدورا ودبيان. وتذكر أننا في حال استخدام Lilo سنحتاج لتحديث الإعداد مجدّداً بالأمر /sbin/lilo أو /sbin/lilo -v.

9. أعد تشغيل الجهاز وتمعّن النتائج (إذا كانت الأمور قد سارت على ما يرام).

---

24 يمكن بدل تنفيذ كل هذه الأوامر، تنفيذ الأمر make install، وهو سيتكفل بهذه العملية بأكملها. هذا الأمر سيتكفل أيضاً بضبط محمل الإقلاع، لذا لا داعي للمرور على النقطة التالية أيضاً.

الأنشطة

1. حدّد إصدار نواة لينكس المضمّنة في توزيعتك. تفقّد التحديثات المتاحة آليًّا، سواء في دبيان (عبر apt) أو في فيدورا/ردهات (عبر yum).
2. قم بعمل تحديث لتوزيعتك. تفحّص الاعتماديات المحتملة مع الوحدات الأخرى المستخدمة (سواء pcmcia أو غيرها)، ومع محمّل الإقلاع المستخدَم (سوءا lilo أو grub-legacy أو grub2). يُنصّح بأخذ نسخة احتياطية من بيانات النظام الهامّة (كمستخدمي النظام، وملفات الإعداد المعدّلة) إذا لم يكن لدينا نظام آخر متاح للتجارب.
3. لفرعنا من النواة، لنحدّد أحدث إصدار متاح (راجع http://www.kernel.org) وقم بعمل تثبيت يدويّ متبعاً الخطوات المشروحة في الوحدة. التثبيت النهائيّ اختياريّ ويمكن إهمالُه، أو يمكنك المُتابعة وإنشاء مُدخلة في محمّل الإقلاع لاختبار النواة الجديدة.
4. في حالة توزيعة دبيان، إضافة إلى الخطوات اليدويّة، رأينا أن هناك طريقة خاصة (مُستحسَنة) لتثبيت النواة من مصادرها باستخدام حزمة النواة.

المراجع

مصادر أخرى للمراجع والمعلومات:

[kerb] موقع يوفّر مستودعاً للإصدارات المختلفة لنواة لينكس وتراقيعها.

[lkm] [kera]مواقع تعود لجزء من مجتمع نواة لينكس. تقدّم العديد من موارد التوثيق والقوائم البريديّة لتطوير النواة، إصداراتها والمزايا الجديدة التي تتطور.

[Dbo] كتاب عن الفرع 2.4 لنواة لينكس، ويعرض مكوناتها المختلفة بالتفصيل، من ناحيتي التصميم والتنفيذ. هناك إصدار أول عن الفرع 2.2 للنواة، وتحديث جديد للفرع 2.6 للنواة.

[Pra] مقال يشرح بعض التطورات الرئيسيّة للسلسلة 2.6 لنواة لينكس.

[Mur] [Ker] مشاريع توثيق للنواة، غير كاملة، لكن فيها مواد مفيدة.

[Tan87] [Vah96] [Bac86] بعض النصوص عن المفاهيم والتصميم والتنفيذ لأنوية إصدارات يونكس المختلفة.

[Pro] [Kan] [Zan01] [Skoa] لمعلومات إضافيّة عن محمّلي الإقلاع Lilo و Grub.

مراجع أخرى للمعلومات بالعربيّة:

كتاب "المرجع المختصر في نواة لينكس" - أشرف خلف. ترجمة عن كتاب Linux in a Nutshell - للهاكر والمطوّر Greg Kroah-Hartman.

كتاب "دليل محمّل الإقلاع Grub 2".

الإدارة المحلية

د. جُسِـــپ جُبرا إستيڤَ

مقدمة

من أول مهام المدير إدارة الموارد المحلية للجهاز. بعض هذه النواحي تم تغطيتها بشكل بسيط في كتاب جنو/لينكس[1]. سنغطي في هذا الجزء تلك المهام الإدارية بشكل أعمق، إضافة إلى النواحي المتعلقة بالتخصيص وكفاءة الموارد.

سنبدأ بتحليل عملية بدء تشغيل نظام جنو/لينكس، مما سيساعدنا على فهم البنية الأساسيّة للنظام، وعلاقته بالخدمات العديدة التي يقدمها.

سنتعلم الآن كيف نحصل على نظرة عامّة عن الوضع الحالي للنظام، باستخدام إجراءات وأوامر مختلفة متوفرة لتقييم الأجزاء المختلفة من النظام، سيسمح لنا هذا باتخاذ القرارات الإدارية إذا وجدنا أخطاء أو سوءاً في الأداء أو إذا وجدنا أننا نفتقد أيًّا من الموارد.

من المهام الرئيسيّة للمدير إدارة حسابات المستخدمين، حيث أن أي إعداد للجهاز سيكون مصمّماً للمستخدمين، سنرى كيف يمكننا تعريف حسابات المستخدمين الجديدة، وتحديد إلى أي مستوى يمكن لهم الوصول إلى الموارد.

أما بالنسبة للأجهزة الثانوية في النظام – كالأقراص والطابعات –، فهناك إمكانيات إدارة مختلفة متاحة، سواء عبر الخوادم المختلفة (للطباعة)، أو أنظمة حفظ الملفات المختلفة التي يمكننا التعامل معها، إضافة إلى بعض تقنيات تحسين أداء الأقراص.

سنختبر أيضاً الحاجة لتحديث النظام، وكيف أنه من الأفضل إبقاؤه محدّثاً، وكذلك سنختبر كيفية تثبيت تطبيقات وبرمجيات جديدة، وكيفية جعل هذه البرامج متاحة للمستخدمين. وفي نفس الوقت، سنحلل المشاكل المتعلقة بتنفيذ المهام الموقوتة المحدّدة مسبقاً في النظام.

في الدرس التعليميّ في نهاية هذا الجزء، سنتعلم كيف نقيّم حالة الجهاز، متبعين النقاط التي رأيناها في هذه الوحدة،

---

1 أظن الكاتب يشير هنا إلى كتاب آخر نشرته Free Technology Academy، وهو GNU/Linux Basic، ويمكن الحصول عليه من هذا الرابط:

.http://ftacademy.org/materials/fsm/13#1

وسنشرح بعض المهام الإدارية البسيطة التي وصفناها. سنناقش في هذه الوحدة بعض الأوامر، وبعدها – في الدرس التعليمي – سنتعامل مع بعضها بتفصيل أكثر، مسلّطين الضوء على كيفية عملها والخيارات المتاحة.

1 التوزيعات: مزايا خاصّة

سنحاول توضيح بعض الاختلافات التقنية الصغيرة (والتي تقلّ مع الوقت) في التوزيعات المستخدمة (فيدورا/ردهات ودبيان)، والتي سنتعامل معها بفصيل أكثر أثناء ظهورها في الوحدة.

تغييرات في فيدورا/ردهات أو خواص فريدة فيها:

- تستخدم محمّل الإقلاع Grub 2 (اختصاراً لعبارة Grand Unified Boot Loader)، وله إعداد في الوضع النصي، ويمكن تعديل إعداداته عند الإقلاع، وله مزايا عديدة. العديد من التوزيعات الحاليّة تستخدم Grub 2، لكن بعض الإصدارات القديمة وبعض التوزيعات ما زالت تستخدم Grub-Legacy القديم، كما أن التوزيعات القديمة جدًّا كانت تستخدم Lilo.[2]
- إدارة البدائل. إذا كان هناك أكثر من برنامج مكافئ متوفّر لمهمّة معينة، فيجب تحديد البديل الذي سيُستخدَم عبر مجلّد /etc/alternatives/. تم استعارة هذا النظام من دبيان التي تستخدمه كثيراً في توزيعتها.
- برنامج فحص منافذ TCP/IP المبنيّ على xinetd، سنجد ملفات الإعداد المقسّمة لوحدات في /etc/xinetd.d/ لبعض خدمات TCP/IP، إضافة إلى ملف الإعداد /etc/xinetd.conf. في أنظمة يونكس التقليديّة، كان النظام المستخدم لهذا الغرض هو inetd، والذي كان له ملف إعداد واحد هو /etc/inetd.conf، وهي الحالة التي كانت في توزيعة دبيان مثلاً، والتي كانت تستخدم inted، وتتيح xinetd اختياريًّا.
- بعض مجلدات الإعداد الخاصّة: /etc/profile.d/، الملفات التي تُنفَّذ عندما يفتح مستخدمٌ صدفةً؛ /etc/xinetd.d/، إعدادات بعض خدمات الشبكة؛ /etc/sysconfig/، بيانات إعداد للعديد من نواحي النظام؛ /etc/cron.*/، العديد من المجلدات التي تُحدَّد فيها المهام التي يجب عملها دوريًّا (عبر crontab)؛ /etc/pam.d/، حيث تعرف وحدات الاستيثاق بالاسم PAM: ويتم ضبط صلاحيات خدمة أو برنامج معيّن في كلّ من ملفات PAM؛ /

---

2 تم إعادة كتابة هذه النقطة بالكامل؛لتفاصيل أكثر حول طريقة إعداد Grub 2، راجع كتاب "دليل محمل الإقلاع Grub 2".

/etc/logrotate.d، إعدادات إعادة تدوير بعض التقارير (متى يكون من الضروريّ تنظيف أو ضغط التقارير، إلخ.) للخدمات المختلفة.

- كان هناك مكتبة برمجيات اسمها kudzu، وتقوم بفحص العتاد عند الإقلاع لإيجاد أيّ تغييرات محتملة (في بعض الإصدارات القديمة جدًّا من فيدورا) في الإعداد، وتقوم كذلك بإنشاء العناصر المناسبة في الإعدادات. لكن هذا الأمر تغيّر الآن، حيث حلّت محلها واجهة برمجية التطبيقات هال Hal API التي تحكّم بهذه الناحية[3].

أمّا بالنسبة لدبيان:

- نظام تحزيم خاص مبني على حزم DEB، مع أدوات بمستويات مختلفة للتعامل مع الحزم، مثل: dpkg, apt-get, dselect, tasksel.
- تتبع دبيان معيار FHS – المتعلّق بهيكليّة المجلدات – مع إضافات خاصة في /etc/، مثل: /etc/default/، ملفات إعداد وقيم مبدئيّة لبعض البرامج؛ /etc/network/، نصوص برمجية لإدارة واجهات الشبكة والبيانات؛ /etc/dpkg/ و /etc/apt/، معلومات حول إعدادات أدوات إدارة الحزم؛ /etc/alternatives/، روابط للبرامج المبدئيّة، والتي بها (أو يمكن أن يكون بها) العديد من البدائل المتاحة.
- نظام الإعداد للعديد من حزم البرمجيات باستخدام أداة dpkg-reconfigure. مثل:

  dpkg-reconfigure gdm

  والذي يجعل من الممكن اختيار المدير القادم لـ X، أو:

  dpkg-reconfigure X-window-system

  الذي يسمح لنا بإعداد العناصر المختلفة في X.
- تستخدم إعدادات خدمات TCP/IP عبر inetd، ملف الإعداد في /etc/inetd.conf؛ هناك أداة update-inetdلتعطيل أو إنشاء مدخلات للخدمات.
- بعض مجلدات الإعدادات الخاصّة: /etc/cron.*/ العديد من المجلدات التي تُحدَّد فيها المهام التي يجب عملها

---

3 تم تحديث هذه النقطة.

دوريًّا (عبر crontab)، /etc/pam.d/، حيث PAM وحدات استيثاق.

2 الإقلاع ومستويات التشغيل

من النقاط الأولى الهامة في تحليل الأداء المحليّ لنظام ما هو كيفية عمله في مستويات التشغيل runlevels، والتي تحدد وضع العمل الحاليّ للنظام، والخدمات المقدمة (في المستوى).

الخدمة هي وظيفة يقدّمها الجهاز، وعادة ما تكون معتمدة على مراقِبين (أو عمليات منفّذة في الخلفيّة تتحكّم بطلبات الشبكة، أو أنشطة العتاد، أو البرامج الأخرى التي تقدّم أيّة مهمّة).

يمكن أن تُفعَّل الخدمات أو تُعَطَّل باستخدام نصوص برمجيّة. معظم العمليات القياسيّة المضبوطة عادة في مجلد /etc/ يُتحكَّم بها عبر نصوص برمجية في /etc/init.d/. النصوص البرمجية ذات الأسماء المشابهة للخدمة المرتبطة بها تظهر عادة في هذا المجلّد، وتقبل عادة معامِلات التشغيل والإيقاف. يتم القيام بالأفعال التالية:

بدء الخدمة	/etc/init.d/service start
إيقاف الخدمة	/etc/init.d/service stop
إيقاف الخدمة ثم إعادة بدئها	/etc/init.d/service restart
عرض حالة الخدمة[4]	/etc/init.d/service status

عندما يبدأ تشغيل نظام جنو/لينكس، يتم في البداية تحميل نواة النظام، ثم تبدأ العملية الأولى؛ يطلق على هذه العمليّة اسم innit، وعليها تنفيذ وتفعيل بقيّة النظام، عبر إدارة مستويات التشغيل المختلفة.

> مستوى التشغيل هو بالأساس إعداد البرامج والخدمات التي ستُنَفَّذ للقيام بمهمّة محدّدة.

المستويات الاعتياديّة – رغم أنه قد يكون هناك اختلاف في الترتيب، خاصة في المستويات 2–5 – في جدول الإعداد لفيدورا والمستحسن في معيار LSB، تكون عادة كالتالي:

مستوى	الوظيفة	الوصف

4 إضافة من المترجم. يمكن أيضاً استخدام الأمر service service-name start لبدء الخدمة، و service service-name stop لإيقاف الخدمة، و service service-name status لعرض حالة الخدمة، و service service-name restart لإعادة بدء الخدمة، و service service-name reload لإعاد تحميل ملف الإعداد. وفي الأنظمة الحديثة، يمكن استخدام طريقة Upstart (كما في دبيان) أو systemd (كما في فيدورا) حسب النظام المستخدَم.

التشغيل		
0	إيقاف التشغيل Halt	يوقف تشغيل البرامج والخدمات التي تعمل، ويفصل أنظمة الملفات الفعّالة.
1	وضع المستخدم الوحيد Single user mode	يوقف تشغيل معظم الخدمات، ويسمح فقط للمستخدم الجذر (المدير) بالولوج. مستخدم لمهام الصيانة وإصلاح الأخطاء الحرجة.
2	وضع متعدّد المستخدمين دون شبكة	لا تعمل أيّ من خدمات الشبكة، ويسمح فقط للولوج المحليّ.
3	متعدّد المستخدمين	تشغّل جميع الخدمات باستثناء الخدمات والبرامج الرسوميّة وما يتعلق بنظام X window.
4	متعدّد المستخدمين	لا يُستخدَم عادة؛ عادة ما يكون مماثلاً للرقم 3.[5]
5	متعدد المستخدمين مع X	مثل 3، لكن مع دعم X لولوج المستخدمين (ولوج رسوميّ).
6	إعادة التشغيل	لجميع البرامج والخدمات. يعيد تشغيل النظام.

ومن ناحية أخرى، علينا أن نذكر بأن دبيان تستخدم نموذجاً لا يحوي أي اختلاف بين المستويات 2–5، فكلها تقوم بنفس المهام تماماً (لكن هذا قد يتغير يوماً ما، بحيث تصير هذه المستويات تتبع معيار LSB).

عادة تكون هذه المستويات مضبوطة في أنظمة جنو/لينكس (ويونكس) بنظامين [أو أسلوبين] مختلفين: وهما BSD و System V (والذي يرمز له اختصاراً بكلمة SysV). في حالة فيدورا ودبيان، فنظام System V هو المستخدَم، وهو النظام الذي سنستخدمه، لكن أنظمة يونكس الأخرى وبعض توزيعات جنو/لينكس (مثل سلاكوير Slackware) تستخدم نموذج BSD.

في حالة نموذج مستويات التشغيل لنظام System V، عندما تبدأ عملية init، فإنها تستخدم ملف إعداد يسمّى /etc/inittab لتحديد وضعية التنفيذ التي ستتخذها. يحدّد هذا النظام مستوى التشغيل المبدئيّ (initdefault) عند الإقلاع (والذي يكون عند التثبيت المستوى رقم 5 في فيدورا و رقم 2 في دبيان)، وسلسلة من الخدمات الطرفية التي يجب تفعيلها ليتمكن المستخدمون من الولوج.

بعد ذلك، سيراجع النظام الملفات الموجودة في /etc/rcn.d/ – طبقاً لمستوى التشغيل المحدّد –، حيث n هو الرقم المرتبط بمستوى التشغيل (المستوى المحدّد)، والذي يحوي قائمة بالخدمات التي يُفترَض أن تبدأ أو تنطفئ عندما نقلع بمستوى التشغيل أو نخرج منه. سنجد بداخل المجلد سلسلة من النصوص البرمجية والروابط للنصوص البرمجية التي تتحكم بالخدمات.

5 عادة ما يُترَك هذا المستوى للمستخدِم ليقرر ماذا سيفعل به، أو يُترك لاستخدامات خاصّة، أو تحسّباً ليُستخدَم عند الحاجة لمستوى جديد.

لكل نصّ برمجي رقم مرتبط بالخدمة، مع رمز بدء S أو K الذان يحدّدان إذا كان هذا النصّ البرمجي لبدء تشغيل (start – S) أو لقتل [أو إيقاف] الخدمة (kill – K)، ورقماً يظهر الترتيب الذي ستُنفَّذ به الخدمات.

تساعدنا مجموعة من أوامر النظام في التحكم بمستويات التشغيل، علينا أن نذكر منها:

- النصوص البرمجية التي سبق ورأيناها في /etc/init.d/ تسمح لنا ببدء، أو إيقاف، أو إعادة بدء الخدمات كلًّا على حدة.
- تسمح لنا telinit بتغيير مستوى التشغيل، علينا ببساطة أن نحدد الرقم. على سبيل المثال، علينا أن نقوم بمهمة حرجة في الجذر، في حال لم يكن هناك مستخدمون آخرون يعملون، فيمكننا أن ننفّذ telinit 1 (يمكن أيضاً استخدام S [بدلاً من الرقم 1، حيث يشير حرف S إلى طور المستخدم "الوحيد" Single]) للانتقال إلى مستوى المستخدم الوحيد، ثم يمكننا بعد الانتهاء من العملية تنفيذ telinit 3 للعودة إلى طَور تعدّد المستخدمين. يمكن أيضاً استخدام الأمر init لنفس المهمة، لكن telinit تقدم قليلاً من المعاملات الإضافيّة. على سبيل المثال، إعادة التشغيل الاعتيادية في أحد أنظمة يونكس تُنفَّذ عبر المزامنة sync، في init 6 يجبر أمر المزامنة sync أنظمة الملفات على إخلاء فيض البيانات، ومن ثم نعيد التشغيل في المستوى 6.
- يسمح لنا أمر إيقاف التشغيل shutdown بإطفاء (halt, –h) أو إعادة تشغيل (reboot, –r) النظام. يمكن أن يتم هذا بعد وقت محدّد أو فوراً. هناك أيضاً أوامر إيقاف التشغيل وإعادته لهذه المهام.
- يسمح لنا wall بإرسال رسائل تحذيرية لمستخدمي النظام. وعلى وجه التحديد، يمكن للمدير تحذير المستخدمين من أن الجهاز سيتوقف عن العمل خلال وقت معين. تستخدمه بعض الأوامر مثل shutdown تلقائيًّا.
- يسمح لنا pidof بإيجاد رقم المعرّف المرتبط بعملية ما. يمكننا الحصول على قوائم بالعمليات عبر ps، وإذا كنا نريد التخلص من عملية أو خدمة عبر kill، فسنرسل رقم معرّفها.

هناك بعض التغييرات الصغيرة في التوزيعات، وفيما يتعلق بنموذج بدء التشغيل هي كالتالي:

- فيدورا/ردهات: ليس لمستوى التشغيل 4 استخدام واضح. مجلدات /etc/rcn.d/ موجودة كروابط للمجلدات الفرعية للمجلد /etc/rc.d/، حيث تتمركز الملفات الأساسيّة لبدء التشغيل. المجلدات كالتالي: /etc/rc.d/rcn.d/،

لكن بما أن الروابط موجودة، فلن يلاحظ المستخدم. مستوى الإقلاع المبدئيّ هو 5 في حال بدء التشغيل بوجود X.

الملفات والأوامر المرتبطة ببدء تشغيل النظام موجودة في حزم البرمجيات sysvinit و initscripts.

فيما يتعلّق بالتغييرات في الملفات والنصوص البرمجية في فيدورا، علينا أن نشير إلى أنه يمكننا إيجاد الملفات التي تحدد القيم المبدئية لإعداد الأجهزة والخدمات في /etc/sysconfig/. النص البرمجي /etc/rc.d/rc.sysinit يُستحضَر مرّة واحدة عند بدء تشغيل النظام، النص البرمجي /etc/rc.d/rc.local يُستحضَر في نهاية العملية ويخدم في تحديد طرق الإقلاع للجهاز.

البدء الحقيقيّ للخدمات يتم عبر النصوص البرمجية المخزنة في /etc/rc.d/init.d/. هناك أيضاً رابط من /etc/init.d/. إضافة إلى ذلك، تقدم فيدورا بعض النصوص البرمجية المفيدة للتحكم بالخدمات: /sbin/service لإيقاف أو بدء خدمة عبر اسمها، و /sbin/chkconfig لإضافة روابط للملفات S و K الضروريّة لخدمة ما أو للحصول على معلومات عن الخدمات.

- لدبيان أوامر إدارة لمستويات الإقلاع مثل update-rc.d الذي يسمح لنا بتثبيت وحذف الخدمات عبر إقلاعها أو إيقافها في واحدة أو أكثر من مستويات التشغيل، يسمح invoke-rc.d بالعمليات التقليديّة لبدء تشغيل وإيقاف وإعادة تشغيل خدمة.

مستوى الإقلاع المبدئي في دبيان هو 2، لا يُدار نظام النوافذ X Window System من /etc/inittab، عوضاً عن ذلك، هناك مدير (مثل gdm أو kdm [أو غيرها]) يعمل كما لو كان خدمة أخرى لمستوى الإقلاع 2.

3 مراقبة حالة النظام

من المهام الرئيسيّة اليومية لمدير النظام (الجذر) التحقق من أن النظام يعمل كما ينبغي والتحقق من وجود أيّة أخطاء محتملة أو استخداماً كبيراً لأيّ من موارد النظام (الذاكرة، الأقراص، إلخ.). في الأقسام الفرعيّة التالية، سندرس الطرق الأساسيّة لاختبار حالة النظام في وقت معيّن وكيفية القيام بالعمليات المطلوبة لتجنّب أيّ مشكلات مستقبلية.

في الدرس التعليميّ في نهاية هذه الوحدة، سنشرح اختباراً كاملاً لنظام افتراضيّ، وبهذا يمكن أن نرى بعضاً من هذه التقنيات.

3.1 إقلاع النظام

عند إقلاع نظام جنو/لينكس، يتم إخراج كمّ ضخم من البيانات المفيدة، عندما يقلع النظام، تظهر الشاشة عادة البيانات من العمليات التي تتفحص خواص الجهاز، والأجهزة التي فيه، وخدمات النظام التي تعمل مع الإقلاع، وغيرها، وتُذكر أيّة أخطاء تظهر أثناء عمل ذلك.

في معظم التوزيعات، يمكن رؤية ذلك مباشرة من محثّ النظام أثناء عملية الإقلاع. لكن سرعة عرض الرسائل، أو بعض التوزيعات الحديثة التي تخفي الرسائل في إقلاع رسوميّ يمكن أن تمنعنا من رؤية الرسائل بشكل جيّد، وهذا يعني أننا نحتاج مجموعة من الأدوات لهذه العلمية.

يمكننا في البداية استخدام ما يلي:

- أمر dmesg: يظهر الرسائل من آخر إقلاع للنواة.
- ملف /var/log/messages: تقرير عام للنظام يتكون من الرسائل التي تنشئها النواة وبعض المراقبات (قد يكون هناك الكثير من ملفات التقارير المختلفة، وعادة ما تكون في /var/log/، وتعتمد على إعداد خدمة syslog.
- الأمر uptime: يعرض المدّة التي بقي فيها النظام يعمل.
- نظام /proc/: نظام ملفات وهمي (procfs) يستخدم النواة لتخزين العمليات ومعلومات النظام.
- نظام /sys/: نظام ملفات وهمي (sysfs) يظهر في فرع النواة 2.6 لتقديم طريقة مفهومة أكثر للوصول إلى معلومات عن الأجهزة ومشغّلاتها.

3.2 النواة: مجلد /proc/

تنشئ النواة عند الإقلاع نظام ملفات وهميّ يسمّى /proc/، وترمي فيه أثناء عملها المعلومات المجمّعة من الجهاز، إضافة إلى الكثير من المعلومات الداخلية. يبقى المجلد /proc/ في الذاكرة، ولا يُحفَظ في القرص. المعلومات التي في داخله تشمل بيانات ثابتة وأخرى متغيرة (وتختلف من تشغيل لآخر).

علينا أن نتذكر أن البنية تعتمد على نواة النظام والبنية المضمّنة، ويمكن أن تتغير الملفات، حيث يعتمد /proc/ كثيراً على النواة.

من النقاط المهمة أنه يمكن أن نجد صور العمليات المنفّذة في مجلد /proc/، إضافة إلى البيانات التي تتلقاها النواة من العمليات. يمكن إيجاد أيّ من عمليات النظام في مجلد /proc/<process PID>/ حيث هناك مجلد بالملفات التي تعرض حالتها. هذه المعلومات أساسيّة لتصحيح البرامج ولأوامر النظام نفسه مثل ps و top، والتي يمكننا أن نستخدمها لرؤية حالة العمليات. وبشكل عام، العديد من أدوات النظام ترجع إلى المعلومات المتغيرة للنظام من /proc/ (وخاصة بعض الأدوات المتوفرة في حزمة procps).

وكملاحظة أخرى، يمكننا أن نجد الملفات الأخرى التي تعرض الحالة العامة للنظام في /proc/. سننظر إلى بعض الملفات التي يمكننا تفحصها للحصول على معلومات هامة بشكل موجز:

الملف	الوصف
/proc/bus	مجلد يحوي معلومات عن منافذ PCI و USB.
/proc/cmdline	ملف بدء النواة *Kernel startup line*
/proc/cpuinfo	معلومات المعالج المركزي CPU
/proc/devices	قائمة بأجهزة المحارف وأجهزة كتب البيانات في النظام
/proc/drive	معلومات عن بعض وحدات النواة للعتاد
/proc/filesystems	أنظمة الملفات المفعّلة في النواة
/proc/ide	مجلد معلومات عن منافذ IDE وخصائص الأقراص
/proc/interrupts	خارطة بطلبات مقاطعات العتاد (IRQ) المستخدمة
/proc/ioports	منافذ الدخل/الخرج المستخدمة I/O ports.

/proc/meminfo	بيانات عن استخدام الذاكرة
/proc/modules	وحدات النواة
/proc/mounts	أنظمة الملفات المضمومة حاليًّا
/proc/net	مجلد بكل معلومات الشبكة
/proc/scsi	مجلد بأجهزة SCSI أو أجهزة IDE التي تحاكى بـ SCSI
/proc/sys	الوصول إلى معامِلات النواة التي يمكن ضبطها بشكل متغير (ديناميكي).
/proc/version	إصدار وتاريخ النواة

في إصدار النواة 2.6، بدء الانتقال من procfs (أي /proc/) إلى sysfs (أي /sys/)، وذلك لنقل كافة البيانات التي لا تتعلق بالعمليات، وخاصة الأجهزة ومشغّلاتها (وحدات النواة) إلى نظام /sys/.

3.3 النواة: مجلد /sys/

نظام sys مسؤول عن إنشاء معلومات عن الأجهزة والمشغلات الموجودة في النواة والمتاحة لمساحة المستخدِم، وبهذا يمكن لواجهات برمجة التطبيقات APIs والتطبيقات الوصول إلى معلومات عن الأجهزة (أو مشغلاتها) بطريقة مرنة أكثر. عادة ما تستخدمها طبقات مثل HAL وخدمة udev لمراقبة الأجهزة وضبطها آليًّا.

يوجد في مفهوم sys هيكلية شجرية لبيانات الأجهزة والمشغلات (لنقل بأنه نموذج مفاهيميّ محدّد)، وكيف يمكن أن يتم الوصول إليها عبر نظام ملفات sysfs (والتي يمكن أن تتغير هيكليتها بين الإصدارات المختلفة).

عند اكتشاف جزء مضاف أو ظهوره في النظام، يتم إنشاء مجلد في sysfs في النموذج الشجري للمشغلات (المشغلات، والأجهزة بما فيها تصنيفاتها المختلفة). تظهر علاقات نقاط الأب/الابن في المجلدات الفرعية في /sys/devices/ (وتعكس الطبقة الفيزيائيّة ومعرفاتها). وتوضع روابط منطقية في المجلدات الفرعية من /sys/bus/ والتي تعكس كيف تنتمي الأجهزة إلى المنافذ الفيزيائية المختلفة في النظام. والأجهزة معروضة في /sys/classes/ ومجمّعة طبقاً لتصنيفها، مثل network لأجهزة الشبكة، و /sys/block/ يحوي أجهزة كل البيانات.

بعض المعلومات التي يقدمها /sys/ يمكن أيضاً إيجادها في /proc/، ولكن هذه الطريقة تم اعتبارها أنها تضمنت خلط العناصر المختلفة (الأجهزة، العمليات، البيانات، العتاد، معاملات النواة) بطريقة ليست منطقية جدًّا، وقد كان هذا أحد أسباب إنشاء /sys/. من المتوقع أن تُرحَّل المعلومات من /proc/ إلى /sys/ لتجميع بيانات الأجهزة فيها.

3.4 العمليات

العمليات التي تكون في حيّز التنفيذ في وقت معيّن تكون بشكل عام ذات طبيعة مختلفة. يمكننا أن نجد:

- عمليات النظام: سواء كانت عمليات مرتبطة بالعمل المحلي للجهاز، أو النواة، أو عمليات (تعرف بالاسم "مراقبات" daemons) معدّة للتحكم بالخدمات المختلفة. ويمكن لهذه العمليات أن تكون محلية أو مرتبطة بشبكة، وهذا يعتمد على ما إذا كانت هذه الخدمات تقدَّم عبر الشبكة (جهازنا يتصرّف كخادم) أو إذا كنا نستقبل نتائج الخدمة (يتصرف جهازنا كعميل). ستظهر معظم هذه العمليات مرتبطة بالمستخدم الجذر، حتى وإن لم نكن موجودين في تلك اللحظة كمستخدمين. قد تكون هناك خدمات أخرى مرتبطة بحسابات نظام أخرى (lp, bin, www, mail، إلخ)، وهم مستخدمون وهميون غير تفاعليين يستخدمهم النظام لتنفيذ عمليات محدّدة.
- عمليات المستخدم المدير: عند العمل في حساب الجذر، ستظهر عملياتنا التفاعلية أو التطبيقات المشغّلة كعمليات مرتبطة بالمستخدم الجذر.
- عمليات مستخدمي النظام: مرتبطة بتنفيذ تطبيقاتها، سواء كانت تطبيقات تفاعلية في الوضع النصي أو الرسومي.

يمكننا استخدام التالية حيث أنها أسرع وأكثر فائدة:

- ps: الأمر القياسيّ، يعرض قائمة بالعمليات، مع بيانات المستخدمين والوقت ومعرفات العمليات وسطر الأوامر المستخدَم. من أكثر الخيارات المستخدمة شيوعاً ps -ef (أو ps -ax)، لكن هناك الكثير من الخيارات المتاحة (راجع دليل الاستخدام man ps).
- top: إصدار يقدم لنا قائمة تحدّث تلقائيًّا كل فترة زمنية قصيرة، ويراقب التغييرات تلقائيًّا. ويسمح لنا بطلب قائمة العمليات مرتبة في تصنيفات مختلفة، مثل استخدام الذاكرة، استخدام المعالج، وهذا من أجل الحصول على تصنيف للعمليات التي تأخذ كل الموارد. إنها مفيدة جدًّا في تقديم معلومات عن المصدر المحتمل للمشكلة، في المواقف التي

تُستَخدَم فيها موارد النظام بشكل كبير.

- kill: يسمح لنا هذا الأمر بالتخلص من عمليات النظام بإرسال أوامر للعملية مثل kill -9 *pid_of_process* (حيث يشير الرقم 9 إلى إشارة SIGKILL)، حيث نحدّد معرف العملية. هذه مفيدة للعمليات ذات التصرف غير المستقر أو التطبيقات التفاعلية التي توقفت عن الاستجابة. يمكننا رؤية قائمة بالإشارات الصالحة في النظام باستخدام man 7 signal.

3.5 سجلات النظام

يمكن للنواة والعديد من مراقبات الخدمات – إضافة إلى الأنظمة الفرعية وتطبيقات جنو/لينكس – إنشاء رسائل تُرسَل إلى ملفات التقارير (أو السجلّات – logs)، سواء للحصول على معلومات حول مسار عمل النظام، أو لاكتشاف أخطاء أو تحذيرات خطأ أو مواقف حرجة. هذا النوع من التقارير ضروريّ في العديد من حالات المهام الإدارية، وكثير من وقت المدير يُقضى في معالجتها وتحليل محتواها.

> تحفظ معظم التقارير في المجلد /var/log/ لكن بعض التطبيقات قد تغير هذا السلوك؛ معظم تقارير النظام نفسه موجودة في هذا المجلد.

هناك مراقب هام في النظام وهو syslogd. مهمة هذا المراقب استقبال البيانات المرسلة من النواة ومراقبات الخدمات الأخرى وإرسالها إلى ملف تقرير موجود في /var/log/messages/. هذا هو الملف المبدئيّ، لكن يمكن ضبط syslogd أيضاً (في الملف etc/syslog.conf/) بحيث يصير من الممكن إنشاء ملفات أخرى اعتماداً على المصدر – طبقاً للمراقب الذي يرسل الرسالة –، وبهذا يتم إرسالها إلى التقرير أو إلى مكان آخر (يحدّد من المصدر)، و/أو فرز الملفات حسب أهميتها (مستوى الأولوية): alarm, warning, error, critical، إلخ.

تبعاً للتوزيعة، يمكن ضبطها بأوضاع مختلفة مبدئيًّا؛ من الممكن في /var/log/ في دبيان إنشاء ملفات مثل: kern.log, mail.err, mail.info, ... وهي تقارير خدمات مختلفة. يمكننا اختبار الإعداد لتحديد المكان الذي تأتي منه الرسائل، وفي أيّ ملفات تُحفَظ. ومن الخيارات التي عادة ما تكون مفيدة إمكانية إرسال الرسائل إلى محثّ نصي افتراضي (في etc/syslog.conf/ يُحدَّد المحثّ الهدف – مثل dev/tty8/ أو dev/xconsole/ – بحسب نوع أو أنواع الرسالة). عادة ما يكون هذا مفيداً في مراقبة تنفيذ النظام دون الاضطرار لتفقّد ملفات التقارير في كلّ حين. وهناك تعديل بسيط للطريقة، حيث يمكن إدخال التعليمات التالية – من طرفية – (للتقرير العام):

```
tail -f /var/log/messages
```

تسمح لنا هذه الجملة بترك الطرفية أو نافذة الطرفية بحيث تظهر التغييرات التي تحصل على الملف تلقائيًّا.

أوامر أخرى ذات علاقة:

- uptime: الوقت الذي بقي فيه النظام عاملاً. مفيدة للتأكد من عدم حدوث إعادة تشغيل غير متوقعة للنظام.
- last: يحلّل التقارير الداخلة والخارجة من النظام (var/log/wtmp/) أو المستخدمين، وإعادة تشغيل النظام. أو التحكم في تقارير last لآخر مرة تمت فيها رؤية المستخدم في النظام (المعلومات في var/log/lastlog/).
- أدوات عديدة للمعالجة المركّبة للتقارير، والتي تنشئ ملخصات (أو تنبيهات) لما حدث في النظام، مثل: logwatch، و logckeck (في دبيان)، و loganalysis (في دبيان أيضاً) ...

3.6 الذاكرة

عندما يتعلق الأمر بذاكرة النظام، علينا أن نتذكر أن لدينا: أ) الذاكرة الفيزيائيّة للجهاز نفسه. ب) الذاكرة الافتراضية التي يمكن للعملية الوصول إليها. عادة (ما لم نكن نتعامل مع خوادم شركات)، لن يكون لدينا قدر كبير جدًّا منها، وبهذا تكون الذاكرة الفيزيائيّة أكثر من الذاكرة الافتراضية الضروريّة (4 جيجا في أنظمة 32بت). سيضطرنا هذا إلى استخدام مناطق التبديل swap على القرص لتنفيذ العمليات المرتبطة بالذاكرة الافتراضية.

يمكن لمنطقة التبديل هذه أن تنفَّذ في ملف في نظام الملفات، لكن إيجادها كقسم swap يُنشأ أثناء عملية تثبيت النظام أكثر شيوعاً. لاختبار معلومات الذاكرة، لدينا العديد من الطرق والأوامر المفيدة:

- ملف etc/fstab/: يظهر قسم التبديل swap (إذا كان موجوداً). يمكننا عبر أمر fdisk إيجاد حجمه (أو بتفقد proc/swaps/).
- الامر ps: يسمح لنا بمعرفة العمليات التي لدينا، مع خيارات عن نسبة وكمّ الذاكرة المستخدمة.
- أمر top: إصدار متغيّر من ps قابل للتحديث خلال مدّة زمنية معينة. يمكن له فرز العمليات حسب استخدامها للذاكرة أو وقت المعالج.
- أمر free: يعرض تقريراً عن الحالة العامة للذاكرة. يقدم أيضاً حجم الذاكرة الافتراضيّة.
- أمر vmstat: يعرض تقريراً عن حالة الذاكرة الافتراضيّة والاستخدام المرتبطة به.
- بعض الحزم – مثل dstat – تسمح لنا بفرز بيانات بمعاملات مختلفة (الذاكرة، و swap، وغيرها) خلال مدّة زمنية قصيرة (مثل top).

3.7 الأقراص وأنظمة الملفات

سنبحث في أيّ الأقراص متاحة، وكيفية ترتيبها، وما الأجزاء وأنظمة الملفات التي لدينا.

عندما يكون لدينا قسم ونظام ملفات معيّن يمكن الوصول إليه، فسيكون علينا القيام بعملية وصل له، وذلك ليتم تضمينه في النظام، سواء تمّ ذلك لهذه المرة فقط، أم تم برمجة هذا الضبط ليتم في كلّ بدء تشغيل/إقلاع. في عملية الوصل، نوصل نظام الملفات ذا العلاقة إلى نقطة في شجرة المجلدات.

للحصول على معلومات عن الأقراص (أو أجهزة التخزين) الموجودة في النظام، يمكننا استخدام معلومات إقلاع النظام (dmesg)، عند اكتشاف المتاح منها، مثل dev/hdx/ لأجهزة IDE أو dev/sdx/ لأجهزة سكَزي SCSI. الأجهزة الأخرى، مثل الأقراص الصلبة المتصلة عبر USB، وأقراص فلاش، والوحدات القابلة للإزالة، وقارئات الأقراص الضوئيّة الخارجية، وغيرها، يمكن أن تكون أجهزة بنوع من محاكاة SCSI، وبهذا ستظهر كأجهزة من هذا النوع.

سيعرض أي جهاز تخزين مجموعة من أجزاء المساحة. وعادة ما يدعم قرص IDE حدًّا أقصى قدره أربعة أقراص فيزيائيّة، أو أكثر إذا كانت منطقيّة (تسمح هذه الأقراص بوضع العديد من الأقسام من هذا النوع على قرص فيزيائيّ واحد)[6]. يمكن لكل قسم احتواء نظام ملفات ذي نوع مختلف، سواء كانت على نفس المشغّل أو على مشغلات مختلفة.

لمعرفة هيكلية جهاز معروف أو لتغيير هيكليته بتقسيم القرص، يمكننا استخدام أمر fdisk أو أيٍّ من أشكاله التفاعلية (cfdisk, sfdisk). على سبيل المثال، عند اختبار قرص ide كمثال، يعرض لنا هذه المعلومات:

---

6 هذا يعتمد على نظام التقسيم المستخدم. في نظام تقسيم Intel/IBM/PC/DOS (الأشهر) الذي يدعمه كل من وندوز ولينكس وماك، لا يمكن إنشاء أكثر من 4 أقسام فيزيائيّة (الحدّ الأقصى هو 4، لكن يمكن الاكتفاء بإثنين أو ثلاثة أو حتى واحد، ولها نوعان: رئيسيّ primary وممتدّ extended)، بحيث يمكن لأحد هذه الأقسام (قسم واحد فقط ويسمّى القسم الممتد extended) أن يحوي عدداً كبيراً من الأقسام المنطقيّة (logical partitions). هناك أنواع أخرى من التقسيم، مثل تقسيم Apple لأجهزة ماك القديمة، وتقسيم Sun Sparc، وتقسيم gpt الأحدث، والذي يقبل عدداً كبيراً من الأقسام، لكن بعض الأنظمة القديمة لا تدعمه، وهو الخيار المبدئي في العديد من التوزيعات الحديثة وفي الإصدارات الحديثة من نظام وندوز.

```
# fdisk -j /dev/hda
```

```
Disk /dev/hda: 20.5 GB, 20520493056 bytes 255 heads, 63
sectors/track, 2494 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id System
/dev/hda1	*	1	1305	10482381	7 HPFS/NTFS
/dev/hda2	*	1306	2429	9028530	83 Linux
/dev/hda3		2430	2494	522112+	82 Linux swap

وهو قرص سعته 20جيجا فيه ثلاثة أقسام (وهي معرّفة بالأرقام المضافة إلى اسم الجهاز)، والذي نرى فيه قسمي إقلاع أحدهما NTFS والآخر من نوع Linux-type (وتعرف بوجود * في عمود الإقلاع Boot)، وهذا يعني وجود وندوز NT/2000/XP/Vistaمع توزيعة جنو/لينكس، وقرصاً أخيراً مستخدماً كمساحة تبديل لينكس. إضافة إلى هذا، فلدينا معلومات عن بنية القرص وحجم كلّ قسم.

بعض الأقراص أو الأقسام التي لدينا، بعضها سيُضَمّ في نظام ملفاتنا، أو ستكون جاهزة للتركيب عند الحاجة، أو يمكن أن تُرَكّب عندما يصبح المورد متوفرّاً (في حالة الأقراص القابلة للإزالة).

يمكننا الحصول على هذه المعلومات بطرق مختلفة (سنرى هذا بمزيد من التفصيل في ورشة العمل النهائيّة):

- يعرض الملف etc/fstab/ الأجهزة الجاهزة للضمّ عند الإقلاع أو الأجهزة القابلة للإزالة التي يمكن ضمّها. لا يشترط أن تظهر كلّ الأجهزة القابلة للإزالة؛ بل فقط تلك التي نريدها أن تظهر عند الإقلاع. يمكننا وصل البقيّة عند الحاجة باستخدام أمر الضمّ أو إزالتها بالأمر umount.
- الأمر mount. يخبرنا هذا الأمر بأنظمة الملفات المضمومة في تلك اللحظة (سواء كانت أجهزة حقيقيّة أو أنظمة ملفات وهميّة مثل /proc/). يمكننا الحصول على هذه المعلومات أيضاً من ملف etc/mtab/.
- الأمر df -k. يخبرنا هذا الأمر بأنظمة ملفات التخزين، ويسمح لنا بالتحقّق من المساحة المستخدمة والمساحة المتاحة. هذا أمر أساسيّ للتحكم بمساحة القرص المتاحة.

فيما يخص الأمر الأخير df -k، فمن المهام الأساسيّة لمدير للنظام التحكّم بموارد الجهاز، وهي في هذه الحالة المساحة المتاحة

في نظام الملفات المستخدم. يجب مراقبة هذه المساحات من وقت لآخر لتجنّب انهيار النظام؛ يجب عدم ترك المساحة الحرّة في نظام الملفات تقلّ عن 10 أو 15% مطلقاً (وخاصّة إذا كانت القرص الرئيسيّ للنظام - الجذر /)، حيث أن هناك العديد من عمليات المراقبات التي عادة ما تكتب معلومات مؤقتة أو تقارير، والتي يمكن أن تُنشئ قدراً كبيراً من المعلومات؛ وهناك حالة معينة، وهي أنّ الملفات الرئيسيّة التي ذكرناها سابقاً يمكن أن تتضمّن ملفات كبيرة جدًّا (وهذا يعتمد على العملية). في العادة، يفترض أخذ بعض الاحتياطات فيما يتعلّق بسلامة النظام، في حال اكتشاف أيّ استخدام كبير غير مبرر لنظام الملفات:

- تخلص من الملفات المؤقتة القديمة. عادة ما يتضمّن المجلدان /tmp/ و /var/tmp/ ملفات كثيرة أنشأها مستخدمون وتطبيقات مختلفة. تقوم بعض التوزيعات والأنظمة بإجراءات سلامة تلقائيّة، مثل تنظيف /tmp/ في كلّ مرة يقلع فيها النظام.
- التقارير logs: بتجنّب النموّ المستمرّ، اعتماداً على ضبط النظام (مثل syslogd)، حيث يمكن أن تكون المعلومات التي تنشئها الرسائل كبيرة جدًّا. في العادة، يجب تنظيف النظام باستمرار، عند استخدام قدر معيّن من المساحة، وعلى أيّة حال، إذا كنّا محتاجين للمعلومات للتحليل التالي، فيمكن عمل نسخ احتياطيّة في أقراص قابلة للإزالة. يمكن أتمتة هذه العملية باستخدام النصوص البرمجية لـ cron أو استخدام أدوات متخصصة مثل logrotate.
- هناك أجزاء أخرى في النظام تنمو كثيراً، مثل: أ) الملفات الأساسيّة للمستخدم: يمكننا حذفها دوريًّا، أو التخلّص من إنشائها؛ ب) نظام البريد الإلكترونيّ: يخزّن كلّ الرسائل المرسلة والمستلمة؛ يمكننا أن نطلب من المستخدمين تنظيفها دوريًّا، أو تنفيذ نظام حصص quota؛ جـ) ذاكرة التخزين المؤقت cache للمتصفحات والتطبيقات الأخرى؛ العناصر الأخرى التي تأخذ كثيراً من المساحة عادة، والتي تتطلّب تنظيفاً دوريًّا، هي: د) حسابات المستخدمين نفسها: يمكن استخدام نظام الحصص معهم، بحيث لا يتم تخطّي المساحة المحجوزة والمحدّدة مسبقاً، إلخ.

4 أنظمة الملفات

في كل جهاز يعمل بنظام جنو/لينكس، سنجد أنواع مختلفة لأنظمة الملفات.

في بادئ الأمر، من الاعتياديّ أن تجد أنظمة ملفات لينكس حقيقيّة مُنشأة في العديد من أقسام الأقراص. الإعداد الاعتياديّ هو وجود قسمين: يعود أحدهما لنظام الملفات الجذر "/" والآخر لمساحة التبديل swap. ولكن، في الإعدادات الأكثر احترافيّة، من المعتاد أن تُفصَل الأقسام في أجزاء "مميّزة" من النظام، ومن التقنيات الشائعة هنا – على سبيل المثال (سنرى خيارات أخرى لاحقاً) – إنشاء أقسام منفصلة بحيث تكون:

/ /boot/ /home/ /opt/ /tmp/ /usr/ /var/ swap

بالتأكيد مضمومة من مصادر مختلفة (أقراص مختلفة، أو حتى الشبكة في بعض الحالات). تقوم الفكرة على فصل الأجزاء الثابتة والمتغيّرة من النظام بشكل واضح، ممّا يسهّل تمديد الأقسام عند ظهور أيّ مشكلة تتعلّق بالتحميل الزائد. أو عزل الأجزاء لتنفيذ عمليات نسخ احتياطيّ بسهولة أكبر (على سبيل المثال، حسابات المستخدمين في قسم /home/).

أقسام التبديل هي أقراص من نوع Linux swap، أما القسم الذي يعود لـ/ فهو نظام الملفات القياسيّ، سواء كان etc2 (وهو ما كان مستخدماً حتى النواة 2.4)، أو الأحدث مثل ext3 أو ext4[7] وهي تحديث لنظام ext2 مع ميزة التسجيل journaling، مما يسمح بوجود سجلّ بما يدخل نظام الملفات، مما يسمح باستعادة أسرع في حال حدوث أخطاء. وهناك أنواع أخرى من أنظمة الملفات، مثل Reiser و XFS، وهي شائعة أيضاً.

وهناك إعداد آخر قد يكون شائعاً وهو استخدام ثلاثة أقسام: وهي /، و swap، و /home/، بحيث يستخدَم /home/ لحسابات المستخدمين. يسمح هذا بفصل حسابات مستخدمي النظام، وذلك بعزل قسمين منفصلين وتحديد المساحة الضروريّة

7 وهناك أيضاً أنظمة ملفات حديثة قد تحلّ محلّ هذه في المستقبل، مثل Btrfs.

للحسابات في قسم آخر.

وهناك إعداد آخر منتشر بكثرة، وهو فصل الأجزاء الثابتة من النظام عن الأجزاء المتغيّرة في أقسام مختلفة، على سبيل المثال، استخدام قسم واحد لوضع / مع الاجزاء الثابتة (وهي في بعض الحالات /bin/ و /sbin/ و /usr/) والتي لا يُتوقَّع لها أن تنمو، وإذا نمت، فلن تنمو بقدْر كبير، وجزء آخر أو أجزاء عديدة أخرى للأجزاء المتغيّرة (/var/ و /tmp/ و /opt/)، على اعتبار أن /opt/ مثلاً نقطة تثبيت البرمجيات الجديدة. هذا يجعل من الممكن ضبط مساحة القرص بشكل أفضل وترك مساحة أكثر لأجزاء تحتاجها من النظام.

حيث تكون أنظمة الملفات المدعومة موصولة، فعلينا التنبيه على تنوّعها، يمكننا حاليًّا إيجاد ما يلي (إضافة إلى غيرها):

- الأنظمة المرتبطة بجنو/لينكس، مثل الأنظمة القياسيّة ext2 و ext3 و ext4، المطوّرة من المفاهيم السابقة للتسجيل (دعم سجلات العمليات المنفَّذة في نظام الملفات والتي تسمح لنا بالاسترجاع في حال وجود أيّ كارثة تجعل البيانات غير مترابطة).
- التوافقية مع بيئات غير جنو/لينكس: MSDOS, VFAT, NTFS، الوصول إلى الأنظمة المختلفة FAT16 و FAT32 و NTFS. وعلى وجه التحديد، علينا أن نذكر بأن النواة – في حالة الاعتماد عليها وحدها – تدعم القراءة فقط [لأنظمة NTFS]. لكن – كما ذكرنا – هناك حلول في مساحة المستخدِم (عبر FUSE، وهي وحدة للنواة تسمح لنا بكتابة أنظمة ملفات تعمل في مساحة المستخدم)، تجعل القراءة/الكتابة ممكنة، مثل NTFS-3g المذكور مسبقاً. هناك أيضاً توافقية مع بيئات أخرى مثل ماك مع HFS و HFSplus.
- الأنظمة المرتبطة بالدعم الفيزيائيّ – مثل الأقراص الضوئيّة CDs/DVDs – مثل أنظمة ISO9660 و UDF.
- أنظمة الملفات المستخدمة في أنظمة يونكس الأخرى، والتي تقدّم أداءاً أفضل (وأحياناً على حساب استهلاك أكبر للموارد، في المعالج على سبيل المثال)، مثل JFS2 (من IBM)، و XFS (من SGI)، و ReiserFS.
- أنظمة ملفات الشبكة (تقليديّة أكثر): NFS، وسامبا (smbfs و cifs)، تسمح لنا بالوصول إلى أنظمة الملفات المتاحة في الأجهزة الأخرى كما لو كانت محلّية باستخدام الشبكة.
- الأنظمة الموزّعة في الشبكة: مثل GFS، و Coda.

- أنظمة الملفات الوهمية، مثل procfs (مجلد /proc/) و sysfs (مجلد /sys/).

في معظم أنظمة الملفات هذه (باستثناء بعض الحالات الخاصّة)، سيسمح لنا جنو/لينكس بإنشاء أقسام من هذه الأنواع، وبناء أنظمة الملفات من النوع المطلوب، وضمّها كجزء داخليّ في شجرة المجلدات، سواء كان هذا بشكل مؤقت أو دائم.

4.1 نقطة الضمّ

بعيدًا عن نظام الملفات الجذر / وأجزائه الإضافيّة الممكنة (/usr/, /var/, /temp/, /home/)، يفترض أن نتذكر أنه من الممكن ترك نقاط الضمّ معدّة لضمّ أنظمة ملفات أخرى، سواء كانت أقساماً من القرص أو أجهزة تخزين أخرى.

في الأجهزة التي يشارك فيها جنو/لينكس الأقسام مع أنظمة تشغيل أخرى، عبر محمّل إقلاع معيّن (مثل lilo او grub)، يمكن أن يكون هناك العديد من الأقسام المرتبطة بأنظمة الملفات المختلفة. عادة ما يكون من الجيد مشاركة بيانات هذه الأنظمة، سواء كان هذا لقراءة أو لتعديل ملفاتها (عكس الأنظمة الأخرى التي تسجّل فقط بياناتها الخاصّة وأنظمة ملفاتها، وفي إصدارات معيّنة منها، بعض أنظمة الملفات هذه لا تكون مدعومة)، فإن لجنو/لينكس القدرة على التعامل مع كمّ هائل من الأنظمة – كما رأينا – من أنظمة تشغيل مختلفة، ومشاركة البيانات معها.

مثال

إذا ثبّتنا نظام جنو/لينكس في الحواسيب الشخصيّة، فسنجد أكثر من نظام تشغيل واحد، فسنجد – على سبيل المثال – إصداراً آخر من جنو/لينكس بنظام ملفات ext2 أو ext3، يمكننا أن نجد نظام MSDOS قديم بنظام FAT التابع لها، ونظام Windows98/ME/XP Home مع FAT32 (أو VFAT للينكس)، أو وندوز NT/2000/XP/Vista مع أنظمة NTFS (نظام NTFS للينكس) و FAT32 (نظام VFAT) في نفس الوقت.

يمكن لنظام جنو/لينكس لدينا قراءة بيانات (وبكلمات أخرى، ملفات ومجلدات) من كلّ أنظمة الملفات هذه والكتابة في معظمها.

في حالة NTFS، وحتى نقطة معينة، [كانت] هناك مشاكل في الكتابة، فقد كانت تجريبيّة في معظم مشغّلات النواة التي ظهرت. وبشكل رئيسيّ بسبب الإصدارات المختلفة لنظام الملفات التي ظهرت على التوالي، وحيث كان هناك إصداران يسميان NTFS و NTFS2، وبعض الامتداد مثل ما يعرف بالأنظمة المتغيّرة، وأنظمة الملفات المشفّرة. وسبّب الوصولُ إليها بمشغلات مختلفة مشاكل عدم توافقيّة معيّنة، مما قد يؤدي إلى تلف في البيانات أو فشل في نظام الملفات.

وبفضل FUSE، وهي وحدة مضمّنة في النواة (منذ الإصدار 2.6.11)، صار من الممكن تطوير أنظمة الملفات بمرونة أكثر، وفي مساحة المستخدم مباشرة (في الحقيقة، يعمل FUSE كـ"جِسر" بين طلبات النواة، والوصول من المشغّلات).

وبفضل مزايا FUSE، صار لدينا دعمٌ كامل لنظام NTFS – إذا افترضنا بأن مايكروسوفت لن تقوم بعمل تغييرات

إضافيّة على معاييره الأساسيّة –، وخاصّة منذ ظهور المشغّل (المبني على FUSE) وهو NTFS-3g (الموقع: http://www.ntfs-3g.org)، ومجموعة أدوات ntfsprogs.

تستخدم أنظمة ملفات مختلفة – اعتماداً على التوزيعة –، أو يمكننا إنشاؤها بأنفسنا. في العادة، توجد إمّا كمجلدات فرعيّة من root، مثل: /cdrom/, /win/, /floppy/، أو كمجلدات فرعيّة بداخل /mnt/، نقطة الضمّ المعياريّة (تظهر بالشكل: /mnt/cdrom/ و /mnt/floppy/ و ...)، أو مجلد /media/ الذي صارت التوزيعات مؤخّراً تُفضّله. بناء على معيار FHS، يفترض أن يُستخدَم /mnt/ لضم أنظمة الملفات مؤقتاً، حيث يفترض أن يُستخدم /media/ لضمّ الأقراص القابلة للإزالة.

يتم تنفيذ عملية الضمّ عبر أمر الضمّ بالهيئة التالية:

```
mount -t filesystem-type device mount-point
```

يمكن أن يكون نوع نظام الملفات: MSDOS (FAT), VFAT (FAT32), NTFS (NTFS read)، أو ISO9660 (للأقراص الضوئيّة المضغوطة CD-ROM) ... إلخ (من الأنظمة الممكنة).

الجهاز هو النقطة في مجلد /dev/ التي تعود على مكان الجهاز، حيث يكون لأجهزة IDE تسميات /dev/hdxy، حيث x تكون أحد الحروف الأربعة a, b, c, d (وهي master و slave، الأول والثاني من كلّ منهما)، يليها رقم القسم [مكان y]، أما أقراص SCSI (وهي /dev/sdx) حيث x يحلّ محله الحروف a, b, c, d, e ... إلخ (اعتماداً على رقم تعريف سكَزي، 1، 2، 3، 4، 5، ...).

سنرى بعض الأمثلة:

```
mount -t iso9660 /dev/hdc /mnt/cdrom
```

هذا سيضمّ القرص الضوئيّ المضغوط CD-ROM (إذا كان قرص IDE متصل بوصلة IDE الرئيسيّة الثانية Secondary Master) في النقطة /mnt/cdrom/.

mount -t iso9660 /dev/cdrom /mnt/cdrom

هذا سيضمّ القرص المضغوط CD-ROM، حيث يُستخدم dev/cdrom/ كمرادِف (هو بالأصل رابط للجهاز المتّصل به قارئ الأقراص المضغوطة.

mount -t vfat /dev/fd0H1440 /mnt/floppy

هذا الأمر سيضمّ القرص المرن dev/fd0H1440/. سيكون نوعه محرك أقراص A عالي التركيز (1.44 ميجا)، يمكن أيضاً استخدام dev/fd0/.

mount -t ntfs /dev/hda2 /mnt/winXP

هذا سيضمّ القسم الثاني من أول جهاز IDE من نوع NTFS (وهو :C [بلغة وندوز])، (على سبيل المثال، نظام وندوز XP).

إذا كانت هذه الأقراص مستقرّة في النظام (وبعبارة أخرى، لا تتغير باستمرار)، وكنا نريد استخدامها، فستكون الطريقة الأفضل تضمين معلومات الضمّ بحيث تتمّ أثناء عملية التنفيذ عند إقلاع النظام، عبر إعداد الملف etc/fstab/:

```
# /etc/fstab: Static information on the file system
#
```

#<Sys. files>	<Mount point>	<Type>	<Options>	<dump>	<pass>
/dev/hda2	/	ext3	errors = remountro	0	1
/dev/hdb3	none	swap	sw	0	0
proc	/proc	proc	defaults	0	0
/dev/fd0	/floppy	auto	user,noauto	0	0
/dev/cdrom	/cdrom	iso9660	ro,user,noauto	0	0
/dev/sdb1	/mnt/usb	vfat	user,noauto	0	0

على سبيل المثال، يشمل هذا الإعداد بعض الأنظمة المعياريّة، مثل dev/hda2/، وقسم التبديل الموجود في dev/hdb3/، ونظام proc (والذي يستخدم النواة لحفظ معلوماته). القرص المرن والضوئيّ، وفي هذه الحالة قرص USB من نوع فلاش (الذي اكتُشِف كجهاز سكَزي). في بعض الحالات، يتم تحديد نوع نظام الملفات auto. يسمح هذا بالاكتشاف التلقائيّ لنظام الملفات. إذا لم يكن معلوماً، فالأفضل تحديده في الإعدادات، وخيار noauto يعني أنه لن يتم ضمّه آليًّا دائماً، لكن عند الطلب (أو الوصول).

إذا كانت لدينا هذه المعلومات في الملف، فهذا يسهّل عملية الوصل، حيث ستتم إما عند التنفيذ أو عند الإقلاع أو عند الطلب (noauto). ويمكن تنفيذها ببساطة بطلب ضمّ نقطة الضمّ أو الجهاز:

```
mount   /mnt/cdrom
mount   /dev/fd0
```

على اعتبار أن النظام لديه بقيّة المعلومات.

العملية العكسية، وهي الفصل (إلغاء الضمّ) سهلة، وهي أمر الفصل umount تليه نقطة الضمّ أو الجهاز:

```
umount   /mnt/cdrom
umount   /dev/fd0
```

عند استخدام أجهزة قابلة للإزالة، مثل الأقراص الضوئيّة (أو غيرها)، فيمكن استخدام أمر الطرد eject لاستخراج القرص الفيزيائيّ:

```
eject /dev/cdrom
```

أو فقط eject في هذه الحالة:

```
eject
```

يقوم أمرا mount و umount بوصل وفصل كل الأنظمة المتاحة. يحوي الملف etc/mtab/ قائمة بالأنظمة المضمومة في لحظة معيّنة، والتي يمكن الرجوع إليها، أو يمكن تنفيذ mount دون معامِلات للحصول على هذه المعلومات.

4.2 الصلاحيات

والشيء الآخر الذي علينا التحكّم به في حالة الملفات والمجلدات هو الصلاحيات التي نرغب بإعطائها لكلّ منها، بينما نبقي في بالنا أن كلّ ملف يمكن أن يكون له سلسلة من الصلاحيات: rwxrwxrwx، حيث تعود [أول] rwx [من الجهة اليسرى] لمالك الملف، و [التي تليها] للمجموعة التي ينتمي لها المستخدم، والأخيرة للمستخدمين الآخرين. وفي كل حالة، يمكننا إعطاء صلاحيات قراءة read – r، وكتابة write – w، وتنفيذ execute – x. في حالة المجلدات، يشير التصريح x إلى إمكانيّة الوصول إلى ذلك المجلّد (عبر أمر cd مثلاً).

لتغيير حقوق الوصول إلى مجلد أو ملف، لدينا الأوامر:

- chown: تغيير مالك الملف.
- chgrp: تغيير المجموعة المالكة للملف.
- chmod: تغيير صلاحيات محددة (rwx) للملفات.

تقدّم الأوامر أيضاً خيار R-، والذي ينفّذ الأمر على الملفات والمجلدات الفرعيّة إذا نُفّذ على مجلّد.

5 المستخدمون والمجموعات

في العادة يكون لمستخدمي جنو/لينكس حساب مرتبط بهم (تُحَدَّد فيه بعض بياناتهم وتفضيلاتهم) إضافة إلى قدر محجوز من المساحة على القرص والتي يمكنهم أن يطوروا فيها ملفاتهم ومجلداتهم. هذه المساحة محجوزة للمستخ]م، ويمكن أن يستخدمها المستخدم فقط (مالم تحدِّد الصلاحيات غير ذلك).

ومن بين الحسابات المرتبطة بالمستخدمين، يمكننا أن نجد أنواع مختلفة:

- حساب المدير، بمعرف الجذر root، والذي يُفترَض أن يُستخدَم للعمليات الإدارية فقط. حساب الجذر هو الحساب الذي لديه أكثر صلاحيات ووصولاً كاملاً للجهاز وللملفات الإعداد. ونتيجة لذلك، فلهذا المستخدم أكبر قدرة على التدمير الذي يمكن أن ينتُج عن أي خطأ أو إزالة. يُفضَّل تجنُّب استخدام حساب الجذر، كما لو كان حساب مستخدم آخَر، لذا يُفضَّل أن يُستخدَم فقط للعمليات الإداريّة.
- حسابات المستخدمين: الحسابات العاديّة لأي مستخدمين للجهاز لديهم صلاحيات مقيّدة لاستخدام ملفات حساباتهم ولبعض المناطق المحدّدة (على سبيل المثال، الملفات المؤقتة في /tmp/)، واستخدام الأجهزة المحدّدة التي سُمِحَ لهم باستخدامِها.
- حسابات الخدمات الخاصّة: حسابات lp, news, wheel, www-data ... التي لا يستخدمها أشخاص، بل تستخدمها الخدمات الداخليّة للنظام، والتي تستخدمها بأسماء المستخدمين هذه. بعض الخدمات تُستخدَم أيضاً ضمن حساب الجذر.

يُنشأ حساب المستخدِم عادة بتحديد اسم (أو معرِّف مستخدم)، وكلمة مرور، ومجلد شخصيّ مرتبط (المجلّد).

المعلومات عن مستخدمي النظام موجودة في الملفات التالية:

```
/etc/passwd
/etc/shadow
/etc/group
/etc/gshadow
```

معلومات عن بعض سطور الملف etc/passwd/:

```
juan:x:1000:1000:Juan Garcia,,,:/home/juan:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

حيث إذا ظهرت النقاط العموديّة :: متتالية، فهذا يعني صندوقاً فارغاً:

- juan: معرّف مستخدم النظام.
- x: كلمة مرور المستخدم مرمّزة، إذا كان هناك x، فهذا يعني أنها موجودة في الملف etc/shadow/.
- 1000: رمز المستخدم، والذي يستخدمه النظام كرمز تعريفيّ للمستخدم.
- 1000: رمز المجموعة الرئيسيّة التي ينتمي إليها المستخدِم. معلومات المجموعة في etc/group/.
- Juan Garcia: تعليق، وعادة يكون الاسم الكامل للمستخدِم.
- home/juan/: المجلد الشخصي المرتبط بهذا الحساب.
- bin/bash/: الصدفة التفاعلية التي يستخدمها المستخدِم عند التفاعل مع النظام في الوضع النصي، أو عبر الصدفة الرسومية. وهي في هذه الحالة باش من جنو، وهي الصدفة المستخدمة مبدئيًّا. الملف etc/passwd/ المستخدَم لاحتواء كلمات مرور المستخدمين بهيئة معمّاة، لكن المشكلة كانت أنه يمكن لأي مستخدم رؤية هذا الملف، وفي وقت ما، تم عمل برامج cracks لمحاولة معرفة كلمات المرور مباشرة باستخدام كلمات المرور المعمّاة مباشرة كنقطة بداية (كلمة معمّاة بنظام التعمية/التشفير).

لتجنّب هذا، لم تعد كلمات المرور تُحفَظ في هذا الملف، بل حل محلها x، ليدلّ هذا على أنها موجودة في ملف آخر يمكن أن يقرأه المستخدم الجذر فقط، وهو etc/shadow/، والذي يمكن أن يكون محتواه مشابهاً لما يلي:

والذي يوجد فيه معرف المستخدم، إضافة إلى كلمة المرور المعمّاة. إضافة إلى هذا، فهي تظهر في أماكن مفصولة

بالنقطتين العموديتين ":" :

- الأيام منذ 1/1/1970 حتى آخر يوم تم تغيير كلمة المرور فيه.
- الأيام المتبقية حتى يتم تغييرها (الرقم 0 يعني أنها لن تتغير).
- الأيام التي يجب تغيير كلمة المرور بعدها (وبعبارة أخرى، مهلة التغيير).
- الأيام التي سيتم فيها تحذير المستخدم قبل انتهاء صلاحية كلمة المرور.
- الأيام بعد انتهاء الصلاحيات، والتي سيُعطَّل الحساب بعدها. عدد الأيام بين 1/1/1970 ويوم تعطيل الحساب.
- ومكان محجوز.

سنجد في etc/group/معلومات عن مجموعات المسخدمين:

```
jose:x:1000:
```

حيث لدينا: اسم المجموعة، يليها كلمة مرور المجموعة (يحل محلها x هنا)، ثمّ معرِّف المجموعة، ثم قائمة المستخدمين:

```
group-name : group-password : group-identifier : list-of-users
```

قد تكون قائمة المستخدمين في المجموعة ظاهرة وقد لا تكون كذلك؛ على افتراض أن المعلومات موجودة مسبقاً في etc/

passwd/، وفي العادة لا تكون موجودة في etc/group/. إذا كانت موجودة هناك، فهي تظهر عادة كقائمة بالمستخدمين تفصل بينهم فاصلة ",". قد يكون للمجموعة أيضاً كلمة مرور (رغم أن هذا ليس شائعاً إلى هذا الحدّ)، كما في حالة المستخدم، وهناك أيضاً ملف shadow، وهو: etc/gshadow/.

ومن الملفات الأخرى المهمّة هي تلك الموجودة في المجلد /etc/skel/، والتي تحوي ملفات يتم تضمينها في كلّ حساب مستخدم عند إنشائه. علينا أن نتذكر أنه – كما رأينا في الصدفات التفاعلية – يمكن أن يكون لدينا نصوص برمجية للإعداد يتم تنفيذها عند دخولنا الحساب أو خروجنا منه. الهياكل العظمية "skeletons" التي يتم نسخها في حسابات المستخدمين عند إنشائها محفوظة في مجلد skel. عادة ما يكون المدير مسؤولاً عن إنشاء ملفات مناسبة للمستخدمين، معطية مسارات التنفيذ الضروريّة، ببدء متغيرات

النظام التي تحتاجها للبرمجيات، إلخ.

سنرى الآن مجموعة من الأوامر المفيدة لإدارة المستخدمين (سنذكر وظائفها وسنقوم ببعض الاختبارات في ورشة العمل):

- useradd: إضافة مستخدم إلى النظام.
- userdel: حذف مستخدم من النظام.
- usermod: تغيير مستخدم في النظام.
- groupadd, groupdel, groupmod: نفسها، لكن للمجموعات.
- newusers, chpasswd: هذه مفيدة جدًّا في التثبيتات الكبيرة بكثير من المستخدمين، حيث تسمح لنا بإنشاء العديد من الحسابات من المعلومات المدخلة إلى ملف newusers، أو تغيير كلمات المرور لعدد كبير من المستخدمين (عبر chpasswd).
- chsh: تغيير صدفة دخول المستخدم.
- chfn: لتغيير معلومات المستخدم المعروضة في التعليق في ملف /etc/passwd.
- passwd: لتغيير كلمة مرور مستخدم. يمكن تنفيذه كمستخدم، وبالتالي سيسأل عن كلمة المرور القديمة والجديدة. عند عمل هذا، على حساب الجذر أن يحدّد المستخدِم الذي سيتم تغيير كلمة مروره (عدا ذلك سيتم تغيير كلمة مرور حساب الجذر)، وكلمة المرور القديمة ليست ضروريّة. على الأرجح أن هذا أكثر أمر يستخدمه الجذر، عندما ينسى المستخدمون كلمات مرورهم القديمة.
- su: يتم تغيير الهوية نوعاً ما. يستخدمها كلّ من المستخدمين والجذر لتغيير المستخدم الحاليّ. في حالة الجذر، تُستخدَم بكثرة لاختبار ما إذا كان حساب المستخدِم يعمل بشكل صحيح؛ هناك أشكال مختلفة: su (دون معاملات، تفيد في تبديل المستخدِم إلى الجذر، بعد تحديد الهوية، مما يجعل من الممكن لنا الانتقال إلى حساب المدير - إذا كنا في الأصل في حساب المستخدم - لتنفيذ مهمة ما). عبارة su username (تغيّر المستخدم إلى username، لكن تترك البيئة كما هي، وبعبارة أخرى، في نفس المجلد ...). الأمر su username mandate (الذي يقوم بعمل

استبدال كامل، كما لو أن المستخدم المذكور قد ولج إلى النظام).

فيما يتعلق بإدارة المستخدمين والمجموعات، ما ذكرناه هنا يتعلّق بالإدارة المحلّيّة لجهاز واحد وحيد. في الأنظمة متعدّدة الأجهزة التي تتشارك المستخدمين، يستخدَم نظام إدارة مختلف لمعلومات المستخدمين. هذه الأنظمة، والتي تعرف بشكل عام بالاسم "أنظمة معلومات الشبكة" Network Information Systems، مثل NIS و +NIS، و LDAP، تستخدم قواعد بيانات لتخزين معلومات المستخدمين والمجموعات بشكل فعّال باستخدام خوادم تُخزَّن فيها قواعد البيانات وأجهزة العملاء الاخرى، وهي المكان الذي يتم الرجوع إليه فيها. هذا يجعل من الممكن وجود نسخة واحدة من بيانات المستخدمين (أو نُسَخ متعدّدة متزامنة) وتجعل من الممكن لهم دخول أي جهاز متاح من المجموعة التي تُديرها هذه الأنظمة. وفي نفس الوقت، تتضمّن هذه الأنظمة مفاهيم إضافية عن الهيكليات و/أو النطاقات/الجهاز ومناطق الموارد، مما يجعل من الممكن تقديم الموارد واستخدامها في المؤسسة بشكل مناسب، مع بُنى مؤسساتية ذات أقسام داخلية وطابع خاصّ.

يمكننا أن نتأكد ما إذا كنا في بيئة من نوع NIS بالنظر إلى compat إذا كانت تظهر في سطر كلمة المرور وملف إعداد المجموعات etc/nsswitch.conf/، إذا كنا نعمل مع ملفات محلية، أو nis أو nisplus طبقاً للنظام الذي نعمل عليه. وبشكل عام، لا يتطلب هذا أي تعديلات للمستخدم البسيط، حيث يتم إدارة الأجهزة كما لو كانت عادية، وسيكون هذا أكثر اعتيادية إذا كان يتم مشاركة الملفات عبر NFS مما يجعل الحسابات متاحة، بغض النظر عن الجهاز المستخدَم. ما زال بالإمكان استخدام معظم الأوامر المذكورة أعلاه دون أيّ مشكلة في NIS و +NIS، والتي تتكافأ فيها، باستثناء أمر تغيير كلمة المرور passwd، والذي نستخدم عادة بدلاً عنه الأمر yppasswd (في NIS) أو nispasswd (في +NIS)؛ لكن في العادة يقوم المستخدم بإعادة تسميتها إلى passwd – عبر رابط – مما يعني أن المستخدمين لن يلاحظوا الفرق.

سنلقي نظرة على هذا وعلى طرق أخرى في وحدات إدارة الشبكة.

6 خدمات الطباعة

يتحدر خادم طباعة جنو/لينكس من تفرعات يونكس BSD، سمّي هذا النظام LPD (اختصار line printer daemon). هذا نظام طباعة قويّ جدًّا، لأنه يشمل القدرة على إدارة الطابعة المحلية وطابعات الشبكة على حد سواء. كما ويقدم الخدمة في النظام لكل من عملاء وخوادم الطباعة.

نظام LPD قديم نوعاً ما، حيث يعود لفرع يونكس BSD (أواسط ثمانينيات القرن الماضي). ونتيجة لذلك، يفتقر LPD لدعم الأجهزة الحديثة، حيث أن النظام لم يأخذ بالحسبان نوع الطباعة الموجود هذه الأيام. لم يصمّم نظام LPD كنظام معتمد على مشغلات الأجهزة، حيث جرت العادة لإنتاج طابعات تسلسلية أو متوازية لطباعة المحارف النصية فقط.

في هذه الأيام، يجتمع نظام LPD مع برمجية شائعة أخرى، مثل نظام Ghostscript الذي يقدّم خرجاً من نوع postscript لنطاق واسع من الطابعات التي يتوفّر لديه المشغل الصحيح لها. وفي نفس الوقت، تضاف إليها برمجية ترشيح تختار المرشّح الصحيح بناء على نوع الملف الذي سيُطبَع. الإجراء الذي يفترض أن يتبع في العادة (هو بالأساس):

- يبدأ العمل بأمر في نظام LPD.
- يحدّد نظام الترشيح نوع الوظيفة (أو الملف) التي يجب أن تُستخدَم ويحوِّل الوظيفة إلى ملف خرج postscript، وهو الملف الذي يتم إرساله إلى الطابعة. في جنو/لينكس ويونكس، تفترض معظم التطبيقات أن الوظيفة ستُرسَل إلى طابعة postscript، وكثير منها يُنشئ خرج postscript مباشرة، ولهذا نحتاج للقيام بالخطوة التالية.
- على Ghostscript أن يُفسِّر ملف postscript الذي يستقبله، ويقوم بالتحويل إلى الصيغة الخاصة للمشغّل اعتماداً على مشغّل الطابعة الذي أرسِل إليها الملف. إذا كانت الطابعة من نوع postscript، فعملية الطباعة مباشرة؛ إذا لم تكن كذلك، فعليه أن "يُترجِم" الوظيفة. تُرسَل المهمّة إلى طابور الطباعة.

بعيداً عن نظام الطباعة LPD (الذي نشأ مع يونكس BSD)، هناك أيضاً النظام المسمى System V (وهو بالأصل الفرع الآخر من يونكس System V). في العادة، ولأسباب تتعلق بالتوافقية، تُضمِّن معظم أنظمة يونكس كلي النظامين، وبهذا يُستخدَم أي من الإثنين كنظام رئيسي، والآخر يحاكي الأول. في حالة جنو/لينكس، تحصل عملية مشابهة، اعتماداً على

التثبيت الذي لدينا، قد يكون لدينا فقط أوامر LPD لنظام الطباعة، لكن سيكون أيضاً من الشائع أوامر System V أيضاً. وهناك طريقة سهلة لتحديد النظامين (BSD أو System V) وهي استخدام أمر الطباعة الرئيسيّ (الذي يرسل الوظائف للنظام)، وهو lpr في BSD، و lp في System V.

هذا هو الحال الابتدائيّ في أنظمة طباعة جنو/لينكس، لكن خلال الأعوام الماضية، ظهرت أنظمة أكثر تُقدِّم مرونة أكثر، وتجعل مشغّلات أكثر متاحة للطابعات. النظامان هما CUPS، والأقل شهرة LPRng. في الحقيقة، لقد صار CUPS معيار جنو/لينكس الأساسيّ، رغم أنه يجب أن تبقى الأنظمة الأخرى مدعومة للتوافقية مع أنظمة يونكس الموجودة.

كلاهما (أي CUPS و LPRng) أنواع لأنظمة ذات مستوى أعلى، لكن الفروق بينهما ليست واضحة لهذه الدرجة بالنسبة للمستخدمين العاديين، إذا ما قورنا بالأنظمة المعيارية لنظامي BSD و System V؛ على سبيل المثال، تُستخدَم نفس أوامر العميل (أو أوامر متوافقة في الخيارات) للطباعة. هناك فروق ملحوظة للمدراء، لأن أنظمة الإعداد مختلفة. وبطريقة ما، يمكننا اعتبار LPRng و CUPS معماريتين جديدتين لأنظمة الطباعة متوافقتين بالنسبة للمستخدمين مقارنة مع الأوامر القديمة.

في توزيعات جنو/لينكس الحالية، يمكننا إيجاد أنظمة طباعة مختلفة. إذا كانت التوزيعة قديمة [جدًّا]، فستحوي فقط نظام BSD LPD؛ في التوزيعات الحديثة: دبيان وفيدورا/ردهات كلتاهما تستخدمان CUPS. في الإصدارات الأقدم لردهات، كانت هناك أداة Print switch جعلت من الممكن تغيير النظام، والتبديل بين أنظمة الطباعة، لكن صار CUPS وحده متاحاً في الآونة الأخيرة. في دبيان، من الممكن تثبيت كلي النظامين، لكن أحدهما لن يقبل الآخَر: يمكن استخدام أحدهما فقط للطباعة.

في حالة فيدورا كور، نظام الطباعة المبدئيّ هو CUPS (حيث اختفى LPRng منها منذ الإصدار الرابع)، وأداوت التحويل Print switchلم تعد موجودة، حيث لم تعد ضروريّة: يُستخدَم system-config-printer لضبط الأجهزة. تستخدم دبيان مبدئيًّا BSD LPD، لكن من الشائع تثبيت CUPS (ويمكننا أن نتوقّع أن يبقى الخيار المبدئيّ في الإصدارات المستقبلية[8])،

8 ربما بسبب نواة BSD في دبيان؛ فدبيان توفّر عدّة أنوية غير لينكس، منها BSD و Hurd، وهذا يجعلها تبتعد عن بعض التقنيات التي تسبب عدم توافقية معها، مثل CUPS للطباعة و systemd لإدارة النظام من إقلاع وخدمات وما شابه.

ويمكن أيضاً استخدام LPRng. إضافة إلى هذا، علينا أن نتذكر أنه لدينا أيضاً إمكانية التفاعل مع أنظمة وندوز (كما رأينا في وحدة الهجرة) عبر موافيق سامبا، والتي سمحت لنا بمشاركة الطابعات والوصول إلى هذه الطابعات.

فيما يتعلق بكلّ من هذه الأنظمة:

- BSD LPD: هو أحد المعايير القياسيّة في يونكس، وتفترض بعض التطبيقات أن الأوامر ونظام الطباعة سيكونا متوفرين، ولهذا يحاكي كلّ من LPRng و CUPS وظائف وأوامر BSD LPD. نظام LPD قابل للاستخدام، لكنه لا يتيح مجالاً كبيراً للضبط، خاصة عندما يتعلق الأمر بالتحكم بالوصول، ولهذا نُقلت التوزيعات إلى أنظمة أخرى أحدث.
- LPRng: صُمّم في الأساس ليُستبدَل بنظام LPD، ولهذا فمعظم الإعداد مشابه، وهناك فقط بعض الملفات المختلفة.
- CUPS: هو أبرز تفرّع من نظام BSD الأصليّ، وضبطه هو نفسه. تُقدَّم فيه العلومات إلى التطبيقات عن الطابعات المتاحة (كما في LPRng). في CUPS، على كلّ من الخادم والعميل أن تتوفر فيهما برمجية CUPS.

يحاكي كلّ من النظامين أوامر طباعة System V.

بالنسبة للطباعة في جنو/لينكس، فالعديد من النواحي يجب أخذها في الحسبان:

- نظام الطباعة المستخدَم: BSD، أو LPRng، أو CUPS.
- جهاز الطباعة (الطابعة): يمكن أن يكون له اتصال محلي بالجهاز، أو يكون على الشبكة. يمكن أن تكون الطابعة الحالية متصلة بالجهاز باتصالات محليّة، عبر واجهات تسلسلية، أو متوازية، أو USB، إلخ. أو قد تكون ببساطة موجودة على الشبكة، كجهاز آخر، أو مع موافيق ملكية خاصة. يمكن لتلك المتصلة عبر الشبكة عادة أن تتصرف كخوادم طباعة (على سبيل المثال، كثير من طابعات HP الليزريّة هي خوادم BSD LPD)، أو يمكن وصلها إلى جهاز يتصرف كخادم طباعة لها.

- موافيق التخاطُب المستخدمة مع الطابعات أو أنظمة الطباعة: سواء كانت اتصالات TCP/IP مباشِرة (كأن تكون HP مع LPD مثلاً) أو أخرى ذات مستوى عالٍ معتمدة على TCP/IP، مثل IPP (تعتمد CUPS)، أو JetDirect (بعض طابعات HP)، وغيرها. هذا المعامِل هام، حيث سيكون علينا معرفته لنتمكن من تثبيت الطابعة في النظام.
- أنظمة الترشيح المستخدمة: يدعم كلّ نظام طباعة واحداً أو أكثر منها.
- مشغّلات الطابعات: في جنو/لينكس، هناك أنواع قليلة مختلفة؛ قد نذكر – على سبيل المثال – مشغّلات CUPS للنظام أو لأطراف خارجية (على سبيل المثال، توفرها HP و Epson)؛ لبرنامج Gimp لتعديل الصور والرسوم أيضاً مشغّلات محسَّنة لطباعة الصور؛ Foomatic نظام إدارة معرفات يعمل مع معظم الأنظمة (CUPS و LPD و LPRng وغيرها)؛ ومشغلات Ghostscript، وغيرها. تقريباً في كلّ الطابعات، هناك واحد أو أكثر من المشغّلات في هذه التصنيفات.

فيما يتعلق بجزء العميل في النظام، فالأوامر الأساسيّة هي نفسها للأنظمة المختلفة، وهذه هي أوامر نظام BSD (تدعم كلّ الأنظمة محاكاة لهذه الأوامر):

- lpr: تُرسَل الوظيفة إلى طابور الطباعة المبدئيّ (أو المحدَّد)، ومن ثمّ يرسلها مراقب الطباعة lpd إلى الطابور ذي العلاقة ويعطيها رقم وظيفة ليُستخدَم مع الأوامر الأخرى. في الوضع الطبيعي، تُحدّد الطابعة المبدئيّة بمتغيّر النظام PRINTER أو بأول واحدة معرّفة وتُستخدَم واحدة موجودة أو – في بعض الأنظمة – سيُستخدَم طابور lp (كالاسم المبدئيّ).

مثال

مثال على lpr:

```
lpr -P epson data.txt
```

يُرسِل هذا الأمر الملف data.txt إلى طابور الطباعة المرتبط بالطابعة التي حدّدناها على أنها "epson".

- lpq: يسمح لنا هذا الأمر بتفقّد الوظائف في الطابور.

مثال

مثال:

# lpq -P epson				
Rank	Owner	Job Files	Total	Size
1st	juan	15	data.txt	74578 bytes
2nd	marta	16	fpppp.F	12394 bytes

يظهر لنا هذا الأمر الوظائف في الطابور، مع الترتيب المرتبط بها وأحجامها؛ قد تظهر الملفات بأسماء مختلفة، حيث يعتمد هذا على ما إذا كنّا أرسلناها بالأمر lpr أو بتطبيق آخر يمكن أن يغيّر الاسم عند إرسالها، أو إذا تمّ استخدام أيّ مرشّحاتٍ لتحويلها.

- lprm: تخلص من الوظائف من الطابور، ويمكننا تحديد رقم الوظيفة أو المستخدِم لإلغاء هذه العمليات.

مثال

lprm -P epson 15

يحذف الوظيفة ذات الرقم 15 من الطابور.

فيما يتعلق بالناحية الإدارية (في BSD)، فالأمر الرئيسيّ هو lpc؛ يمكن أن يُستخدَم هذا الأمر لتفعيل أو إلغاء تفعيل الطوابير، أو نقل الوظائف في ترتيب الطابور، وتفعيل أو إلغاء تفعيل الطابعات (يُمكن أن تُستقبَل الوظائف في الطابور، لكن لا تُرسَل إلى الطابعات).

علينا أن نذكر أيضاً أنه في حالة System V، فإن أوامر الطباعة في العادة متاحة أيضاً، فعادة ما تُحاكى بناء على أوامر BSD. في حالة العميل، الأوامر هي: lp, lpstat, cancel، وللمهام الإدارية: lpadmin, accept, reject, lpmove, enable, disable, lpshut.

في الأقسام التالية، سنرى أنه من الضروريّ ضبط خادم الطباعة للأنظمة الثلاثة الرئيسيّة. يمكن أن تُستخدَم هذه الخوادم لكلّ من الطباعة المحلية وطباعة عملاء الشبكة (إذا كانت مفعّلة).

6.1 BSD LPD

في حالة خوادم BSD LPD، هناك نوعان رئيسيان من الملفات يجب التعامل معهما: فمن ناحية، تعريف الطابعات في /etc/printcap، ومن ناحية أخرى صلاحيات الوصول عبر الشبكة في /etc/hosts.lpd.

فيما يتعلق بالصلاحيات، فالوضع المبدئيّ هو أن يعطي BSD LPD صلاحيات فقط للوصول المحلي للطابعة، لذا يجب أن تكون مفعلة بشكل مخصوص في /etc/hosts.lpd.

مثال

يمكن أن يكون الملف:

```
# file hosts.lpd
second
first.the.com
192.168.1.7
+@groupnis
-three.the.com
```

مما قد يوحي بأنه من الممكن الطباعة على عدد من الأجهزة، مسرودة إما بأسماء DNS أو بعناوين IP. يمكن إضافة مجموعات الأجهزة المنتمية إلى خادم NIS (وهي groupnis كما في المثال)، أو يمكن حظر الوصول عن أجهزة عديدة بالإشارة إلى ذلك بالشَرطة (-).

فيما يتعلق بإعداد الخادم في /etc/printcap، يمكننا تحديد مدخلات تحدد كل منها طابور نظام طباعة يمكن استخدامه لإيقاف وظائف الطباعة. يمكن أن يكون الطابور مرتبطاً بجهاز محلي أو خادم بعيد، سواء كان طابعة أو خادماً آخر.

يمكن أن توجد هذه الخيارات في أي منفذ:

- lp = : يحدد الجهاز الذي ترتبط به الطابعة، فمثلاً: lp = /dev/lp0 تشير إلى المنفذ المتوازي الأول. إذا كانت الطابعة من نوع LPD، كأن تكون طابعة شبكة تقبل ميفاق LPD (مثل HP)، عندها يمكننا ترك الصندوق فارغاً وملء ما يليه.

- rm = : عنوان باسم أو IP الجهاز البعيد الذي سيستخدم طابور الطباعة. إذا كانت طابعة شبكة، فسيكون عنوان الطابعة نفسها.
- rp = : اسم الطابور البعيد، في الجهاز المعرف مسبقاً بالخيار rm.

فلننظر إلى المثال التالي:

```
# Local printer input lp|epson|Epson C62:\
:lp=/dev/lp1:sd=/var/spool/lpd/epson:\
:sh:pw#80:pl#72:px#1440:mx#0:\
:if = /etc/magicfilter/StylusColor@720dpi-filter:\filter
:af = /var/log/lp-acct:lf = /var/log/lp-errs:
# Remote printer input
hpremote|hpr|remote hp of the department|:\
:lp = :\
:rm = server:rp = queuehp:\
:lf = /var/adm/lpd_rem_errs:\log file.
:sd = /var/spool/lpd/hpremote:local associated spool
```

6.2 LPRng

في حالة نظام LPRng، وللاحتفاظ بالتوافقية مع نظام BSD إضافة إلى تحسينات أخرى تتعلق بالوصول، فالنظام متوافق فيما يتعلق بإعداد الطوابير، ويتم الضبط عبر نفس هيئة الملفات، /etc/printcap، مع بعض العمليات الإضافية الفريدة.

أما الاختلاف في الإعداد فهو فيما يتعلق بالإعداد: في هذه الحالة، نحصل على الوصول عموماً عبر الملف /etc/lpd.perms العام لكل النظام، والذي يمكن أن يكون فيه إعدادات منفردة لكل طابور بتبديل lpd.perms إلى ملف بنفس الاسم في المجلد المرتبط بالطابور، والذي عادة ما يكون في /var/spool/lpd/queue-name/.

ملفات lpd-perms هذه لها سعة أكبر لإعداد الوصول وتسمح بالأوامر الأساسية التالية:

```
DEFAULT ACCEPT
DEFAULT REJECT
ACCEPT [ key = value [,value]* ]*
REJECT [ key = value [,value]* ]*
```

حيث يسمح لنا الأولان بإنشاء قيمة أولية، بقبول أو رفض كل شيء، والآخران يسمحان لنا بقبول أو رفض إعداد معين في السطر. من الممكن قبول أو رفض طلبات من مضيف معين، أو من مستخدم أو منفذ أو عنوان IP. وكذلك أيضاً من الممكن تحديد نوع الخدمة التي ستُقدَّم للعناصِر: X (يمكن أن يكون متصلاً)، P (وظيفة الطباعة)، Q (اختبار الطوابير بالأمر lpq)، و M (لإزالة الوظائف من الطابور، lprm)، و C (للتحكم بالطابعات، الأمر lpc)، إضافة إلى غيرها، كما في الملف التالي:

```
ACCEPT SERVICE = M HOST = first USER = jose
ACCEPT SERVICE = M SERVER REMOTEUSER = root
REJECT SERVICE = M
```

يسمح بحذف وظائف من الطابور لكل من المستخدم first للجهاز، والمستخدم الجذر على الخادم المستضافة عليه خدمة الطباعة (localhost)، إضافة إلى هذا، فإن أيّ طلبات أخرى لحذف وظائف من الطابور لم تُنفَّذ بالفعل ستُرفَض.

علينا أن نكون بكامل حذرنا عند عمل هذا الإعداد، لأن LPRng في بعض التوزيعات تكون مفتوحة مبدئيًّا. الاتصال قد

يكون محدّدًا مثل:

```
ACCEPT SERVICE = X SERVER
REJECT SERVICE = X NOT REMOTEIP = 100.200.0.0/255
```

خدمة الوصول متاحة فقط للجهاز المحلي للخادم وترفض الوصول إذا كان الجهاز غير منتمٍ لشبكتنا الفرعية (في هذه الحالة، نفترض بأن شبكتنا هي 100.200.0.0 – 100.200.0.254).

فيما يتعلق بالإدارة من سطر الأوامر، فلدينا نفس الأدوات القياسيّة المستخدمة في BSD. وفيما يتعلق بالإدارة الرسومية للنظام، فعلينا أن نشير إلى أداة lprngtool (غير متوفرة في جميع إصدارات نظام LPRng).

Names (name|alias1|...) lp ?
Comments ?
Spool Directory /var/spool/lpd/%P ?
Hostname/IP of Printer h14 ?
Port number 9100 ?
IFHP User Specified Filter /usr/libexec/filters/ifhp ?
Select Printer Model and Filter Options default ?
Job Options Select LPR Job and Filter Options landscape ?
Printcap for: ?
Server and Client (BOTH) Server Only (:server) Client Only (:client)
Spool action:
Localhost (:force_localhost) Remote Queue or Device (:force_localhost@) Default
Printer Type: ?
Device Queue TCP/IP Socket SMB/Novell/AppleTalk
Load Balance Dummy Unknown
OK Cancel Advanced Options

شكل 1: lprngtool، ضبط طابعة

هناك العديد من حزم البرمجيات المتعلقة بنظام LPRng؛ فمثلاً، يمكننا في ديبان أن نجد:

lprng – lpr/lpd printer spooling system

lprng-doc – lpr/lpd printer spooling system (documentation)

lprngtool – GUI front-end to LPRng based /etc/printcap

printop – Graphical interface to the LPRng print system.

6.3 CUPS

CUPS معمارية جديدة لنظام الطباعة مختلف نوعاً ما، له طبقة توافقية مع BSD LPD، مما يعني أنه يمكنه التفاعل مع مع خوادم من هذا النوع. ويدعم CUPS أيضاً ميفاقاً جديداً يطلق عليه اسم IPP (يعتمد على http)، لكنه متوفر فقط عندما يكون كل من الخادم والعميل من نوع CUPS. إضافة إلى ذلك، يستخدم CUPS نوعاً من المعرّفات يدعى PPD للتعرّف على إمكانيات الطابعة، يأتي CUPS ببعض هذه المعرفات، كما ويوفّرها بعض المصنّعين (مثل HP و Epson).

[إضافة لهذا،] فله نظام إعداد مختلف كليًّا، ويعتمد على ملفات مختلفة: etc/cups/cupsd.conf/ يجعل ضبط نظام الطباعة مركزياً، و etc/cups/printers.conf/ الذي يتحكم بتعريف الطابعات، و etc/cups/classes.conf/ لمجموعات الطابعات.

يمكننا في etc/cups/cupsd.conf/ إعداد النظام اعتماداً على مجموعة من أجزاء الملف وتعليمات الأعمال المختلفة. الملف كبير نوعاً ما، سنذكر بعض التعليمات الهامة:

- allow : يسمح لنا هذا الخيار بتحديد أي الأجهزة يمكنها الوصول إلى الخادم، سواء ضمن مجموعات أو فُرادى، أو إلى أجزاء من عنوان IP للشبكة.
- AuthClass : تجعل من الممكن تحديد إذا كان سيُطلَب من العملاء المستخدمين استيثاق حسابهم أم لا.
- BrowseXXX : هناك مجموعة من التعليمات المرتبطة بإمكانية تفحّص شبكة لإيجاد الطابعات المخدومة، هذه الإمكانية مفعّلة مبدئيًّا (browsing on)، مما يعني أننا سنجد في العادة أن كل الطابعات الموجودة في الشبكة متاحة. يمكننا تعطيلها بحيث لا نرى إلا الطابعات التي عرّفناها. من الخيارات الأخرى الهامة خيار BrowseAll، والذي نستخدمه لنحدد من المسموح لهم طلب طابعتنا، هذا الخيار مفعل مبدئيًّا، مما يعني بأنه يمكن لأي شخص أن يرى طابعتنا من شبكتنا.

علينا أن نشير إلى أن CUPS في الأساس مصمم بحيث يعمل العملاء والخادم على نفس النظام، إذا كان العملاء يستخدمون LPD أو LPRng، فمن الضروريّ تثبيت مراقب توافقية يسمى cups-lpd (والذي عادة ما يكون في حزم مثل

(cupsys-bsd). في هذه الحالة، يستقبل CUPS الوظائف التي تأتي من أنظمة LPD و LPRng، لكنه لا يتحكم بالوصول (يعمل cupsd.conf لنظام CUPS فقط، ولهذا، فسيكون من الضروري اتخاذ استراتيجية ما للتحكم بالوصول، كجدار ناري مثلاً (انظر إلى الجزء المتعلق بالأمن).

للإدارة من سطر الأوامر، فإن cups فريد نوعاً ما من حيث قبوله لنظامي LPD و System V في العميل، وعادة ما تتم الإدارة بأمر System V وهو lpadmin.

أما فيما يتعلق بالواجهات الرسوميّة، فلدينا gnome-cups-manager، و gtklp، أو واجهة الوِب التي تأتي مع نظام CUPS نفسه، والذي يمكن الوصول إليه على العنوان http://localhost:631.



شكل 2: واجهة لإدارة نظام CUPS

فيما يتعلق بحزم البرمجيات التي تندرج تحت CUPS، فيمكننا أن نجد في دبيان (إضافة إلى غيرها) :

cupsys - Common UNIX Printing System(tm) - server
cupsys-bsd - Common UNIX Printing System(tm) - BSD commands
cupsys-client - Common UNIX Printing System(tm) - client programs (SysV)
cupsys-driver-gimpprint - Gimp-Print printer drivers for CUPS
cupsys-pt - Tool for viewing/managing print jobs under CUPS
cupsomatic-ppd - linuxprinting.org printer support - transition package
foomatic-db - linuxprinting.org printer support - database
foomatic-db-engine - linuxprinting.org printer support - programs
foomatic-db-gimp-print - linuxprinting - db Gimp-Print printer drivers
foomatic-db-hpijs - linuxprinting - db HPIJS printers
foomatic-filters - linuxprinting.org printer support - filters
foomatic-filters-ppds - linuxprinting - prebuilt PPD files
foomatic-gui - GNOME interface for Foomatic printer filter system
gimpprint-doc - Users' Guide for GIMP-Print and CUPS
gimpprint-locals - Local data files for gimp-print
gnome-cups-manager - CUPS printer admin tool for GNOME
gtklp - Front-end for cups written in gtk

7 إدارة الأقراص

فيما يتعلق بوحدات التخزين، فكما رأينا، فلها مجموعة من الأجهزة المرتبطة بها، تعتمد على نوع الواجهة المستخدمة:

- IDE: الأجهزة:

  قرص رئيسي، الوصلة الأولى dev/sda/

  قرص تابع على الوصلة الأولى dev/sdb/

  قرص رئيسي على الوصلة الثانية dev/sdc/

  قرص تابع على الوصلة الثانية dev/sdd/

- SCSI: الأجهزة dev/sda/ و dev/sdb/ ...إلخ. وتتبع ترقيم الأجهزة الطرفية على منفذ SCSI.
- الأقراص المرنة: أجهزة dev/fdx/، حيث يكون x رقم القرص (بدءاً من الرقم صفر). هناك العديد من الأجهزة اعتماداً على سعة القرص، فعلى سبيل المثال، قرص 1.44 ميجا في محرك الأقراص A سيكون dev/fd0H1440/.

فيما يتعلق بأقسام القرص، فالرقم الذي يتبع اسم القرص يحدد رقم القسم في القرص، ويتم التعامل معه كجهاز مستقل: فالجهاز dev/hda1/ هو القسم الاول من قرص IDE الأول، و dev/sdc2 هو القسم الثاني من جهاز SCSI الثالث. في حالة أقراص IDE، فهي تسمح بأربعة أقسام تعرف بالأقسام الرئيسية primary، وعدداً أكبر من الأجزاء المنطقية. ولهذا، إذا كان dev/hdaN/، وكان N أصغر من أو يساوي 4، فسيكون القسم رئيسيًّا، أما إذا لم يكن كذلك، فسيكون قرصاً منطقيًّا، حيث سيكون N أكبر من أو يساوي 5.

فيما يتعلق بالأقراص وأنظمة الملفات المرتبطة بها، فالعمليات الأساسية التي يمكننا القيام بها تندرج تحت:

- إنشاء أو تعديل الأقسام. عبر أوامر مثل fdisk وأشباهه (مثل cfdisk و sfdisk).
- تهيئة الأقراص المرنة: يمكن استخدام أدوات مختلفة للأقراص المرنة: fdformat (للتهيئة منخفضة المستوى)، و superformat (للتهيئة على سعات مختلفة بهيئة MSDOS)، و mformat (تهيئة معينة لإنشاء نظام ملفات

MSDOS قياسي).

- إنشاء أنظمة ملفات لينكس – في الأقسام – باستخدام الأمر mkfs. هناك إصدارات معينة لإنشاء أنظمة ملفات عديدة، مثل mkfs.ext2 و mkfs.ext3، بل وحتى أنظمة ملفات غير لينكساوية، مثل: mkfs.ntfs، و mkfs.vfat، و mkfs.msdos، و mkfs.minix، وغيرها. بالنسبة للأقراص الضوئية، فهناك أوامر مثل mkisofs لإنشاء ISO9660 (بإضافات joliet و rock ridge)، والتي يمكن أن تكون صورة يمكن تسجيلها لاحقاً على قرص CD أو DVD، والتي ستسمح لنا أخيراً بمساعدة أدوات مثل cdrecord بإنشاء وحفظ الأقراص الضوئيّة CD/DVD. وهناك حالة أخرى، وهي الأمر mkswapfs، والذي يسمح لنا بإنشاء مساحات إبدال في الأقسام، والتي يتم تفعيلها أو تعطيلها لاحقاً بالأمرين swapon و swapoff.

- إعداد أنظمة الملفات: الأمران mount و umount.

- التحقق من الحالة: الأداة الرئيسية للتحقق من أنظمة ملفات لينكس هي الأمر fsck. يتفحص هذا الأمر المناطق المختلفة لنظام الملفات للتحقق من سلامتها، ويبحث عن الأخطاء المحتملة ويصححها عندما يكون هذا ممكناً. النظام نفسه يفعل الأمر تلقائيًا عند الإقلاع، وذلك عندما يكتشف مواقف يكون فيها النظام لم يُطفأ بطريقة صحيحة (بسبب انقطاع في التيار الكهربائي، أو إطفاء الجهاز بالخطأ)، أو عند إقلاع النظام لعدد معين من المرات؛ يستغرق هذا الفحص عادة مقداراً معيناً من الوقت، وعادة ما يكون بضع دقائق (اعتماداً على حجم البيانات). هناك أيضاً إصدارات خاصة لأنظمة ملفات مختلفة: fsck.ext2، و fsck.ext3، و fsck.vfat، و fsck.msdos ...إلخ. عادة ما يتم تنفيذ عملية fsck على الجهاز وهو في حالة القراءة فقط والأقراص مضمومة؛ يُنصَح بفصل الأقسام للقيام بالعملية إذا اكتُشفت أخطاء وكان من الضروري إصلاحها. في حالات معينة، كأن يكون نظام الملفات الذي يجب فحصه هو نظام الملفات الجذر (/)، وتم اكتشاف خطأ حرج، فسيُطلَب منا تغيير وضع تنفيذ مستوى تشغيل النظام إلى وضع تنفيذ الجذر، وأن نقوم بعملية التحقق هناك. وبشكل عام، إذا كان من الضروري فحص النظام، فيجب القيام بذلك بوضع المستخدم الجذر (يمكننا التبديل بين مستويات التشغيل بالأمرين init و telinit).

- عمليات النسخ الاحتياطي: سواء في القرص، أو كُتَل من القرص، أو أقسام، أو أنظمة ملفات، أو ملفات ...

فهناك العديد من الأدوات المفيدة لهذا الغرض: يسمح لنا tar بنسخ الملفات إلى ملف أو وحدات أشرطة تخزين، cpio كذلك يمكنه عمل نسخ احتياطي للملفات إلى ملف، يحتفظ كل من cpio و tar بمعلومات عن الصلاحيات ومالكي الملفات، يجعل dd من الممكن عمل نسخ، سواء كانت ملفات، أجهزة، أقسام، أو أقراص إلى ملفات، الأمر معقد قليلاً، ولدينا معلومات ذات مستوى منخفض عن النوع، الحجم، الجزء أو القطاع، ويمكن أيضاً نقلها إلى الأشرطة.

- أدوات مختلفة: بعض الأوامر المنفردة، والتي تستخدم العمليات السابقة بعضها للقيام بالعديد من الإصلاحات: مثل badblocks لإيجاد القطاعات المعطوبة في الجهاز، dumpe2fs للحصول على معلومات عن أنظمة ملفات لينكس، tune2fs يجعل من الممكن القيام بتوليف نظام ملفات لينكس من نوع ext2 و ext3 ضبط معاملات الأداء المختلفة.

سنذكر الآن موضوعين متعلقين بمفهوم مساحة التخزين، وهما مستخدمان في العديد من البيئات للإنشاء الأساسيّ لمساحة التخزين، وهما استخدام برمجيات ريد RAID، وإنشاء الوسائط الديناميكيّة.

7.1 برمجيات RAID

ضبط الأقراص باستخدام مستويات RAID أحد أكثر أنظمة التخزين ذات التواجد الدائم استخداماً وانتشاراً، عندما يكون لدينا العديد من الأقراص لتنفيذ نظام ملفاتنا.

يعتمد التركيز الأساسي على التقنيات المختلفة الموجودة على التسامح مع الأخطاء الذي يقدمه مستوى الجهاز ومجموعة الأقراص إلى العديد من الأخطاء المحتملة، سواء كانت فيزيائية أو في النظام، لتفادي خسارة البيانات أو قلّة الترابط في النظام. إضافة إلى بعض الأنماط المصممة لزيادة أداء نظام الأقراص، بزيادة عرض النطاق لهذه الأقراص المتاحة للنظام والتطبيقات.

يمكننا اليوم إيجاد RAID في العتاد، وبشكل رئيسي في خوادم الشركات (رغم هذا فقد بدأت بالظهور في الأجهزة المكتبية)، حيث هناك حلول عتادية مختلفة متاحة لتلبية هذه المتطلبات. على وجه الخصوص، للتطبيقات التي تستخدم الأقراص بكثرة، كتشغيل الصوتيات والمرئيات، أو في قواعد البيانات الضخمة.

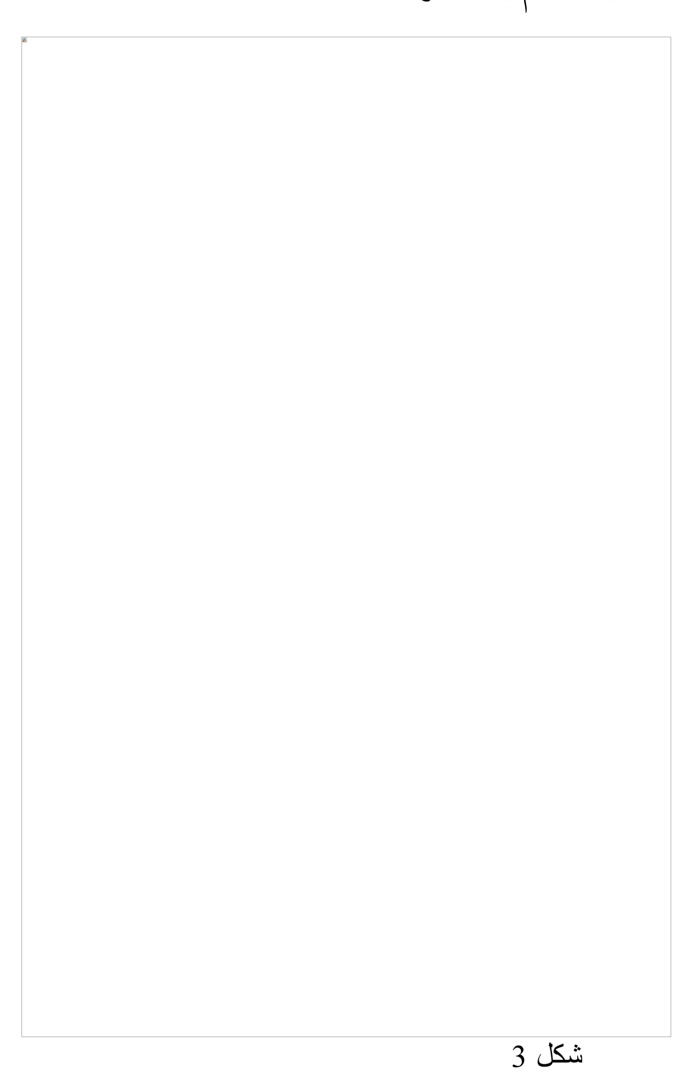
وبشكل عام، يكون هذا العتاد على شكل بطاقات (أو مضمّناً في الجهاز) من مشغلات أقراص من نوع RAID، والتي تقوم بإدارة مستوى واحد أو أكثر (من معيار RAID) على مجموعة من الأقراص المدارة بهذا المشغّل.

يأتي ريد بمجموعة من المستويات (أو الإعدادات الممكنة)، التي يمكن تقديمها (يدعم كلّ مصنع لعتاد معين أو برمجية معينة واحداً أو أكثر من هذه المستويات). يُطبَّق كل مستوى ريد على مجموعة من الأقراص يطلق عليها أحياناً مصفوفة ريد (RAID array أو RAID disk matrix)، والتي عادة ما تكون أقراصاً بسعات متساوية (أو مساوية لأحجام المجموعات). على سبيل المثال، في حالة مصفوفة، يمكن استخدام أربعة أقراص بسعة 100جيجا للقرص، أو يمكن استخدام مجموعتين (بسعة 100 جيجا) لقرصين أحدهما بسعة 30 جيجا والآخر 70. في بعض حالات مشغلات العتاد، لا يمكن للأقراص أو المجموعات أن تكون بأحجام مختلفة؛ في حالات أخرى يمكن ذلك، لكن يتم تحديد حجم المصفوفة بحجم أصغر قرص (أو مجموعة).

سنصف بعض المفاهيم الأساسيّة عن بعض المستويات في القائمة التالية (علينا أن نتذكر أنه – في بعض الحالات – قد

لا تكون كل المصطلحات هي نفسها المستخدمة، وقد تعتمد على كلّ مصنع على حدة):

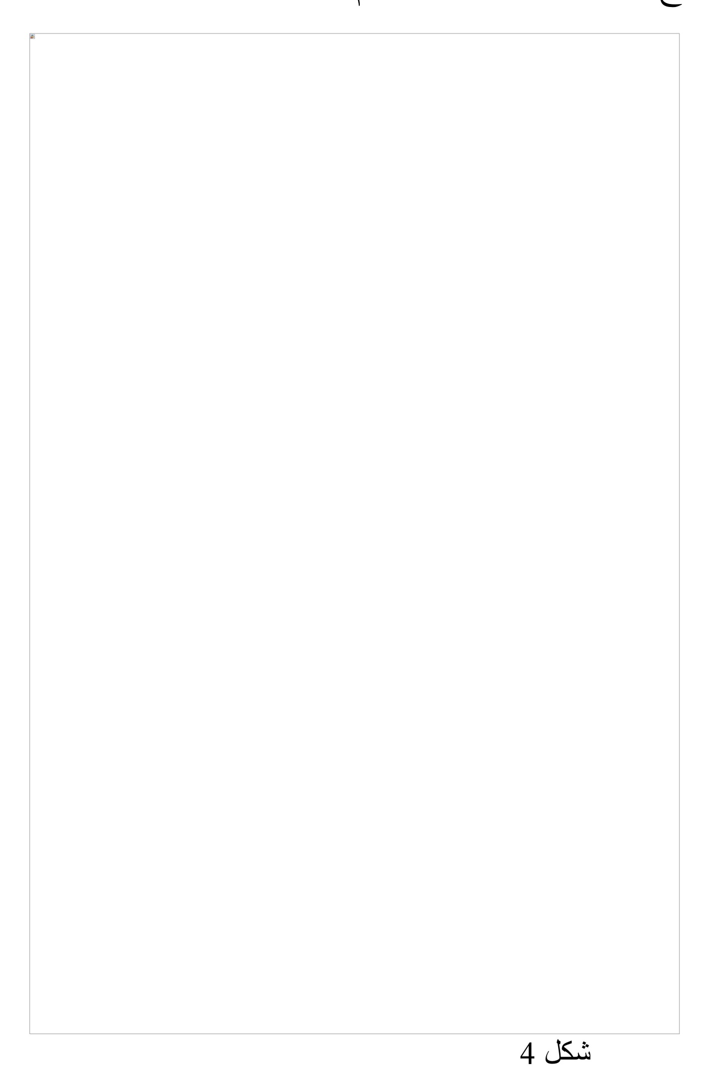
- RAID 0 : البيانات موزعة بالتساوي على واحد أو أكثر من الأقراص دون معلومات للتحقق أو التكرار، دون توفير تسامح مع الأخطاء. فقط البيانات يتم توزيعها؛ إذا فشل القرص فيزيائيًّا، فستُفقَد البيانات، وسيكون علينا استرجاعها من النسخ الاحتياطية. ما يزداد في هذه الحالة هو الأداء، اعتماداً على تنفيذ RAID المستخدَم، آخذين بعين الاعتبار أن خيارات القراءة والكتابة ستُقسَّم بين الأقراص المختلفة.



شكل 3

- RAID 1 : تُنشأ نسخة (مرآة) مطابقة تماماً في مجموعة من قرصين أو أكثر (تعرف باسم مصفوفة ريد). ريد مفيد في هذه الحالة في أداء القراءة (الذي يمكن أن يزداد خطيًّا اعتماداً على عدد الأقراص)، والأهم فائدته في الحصول على

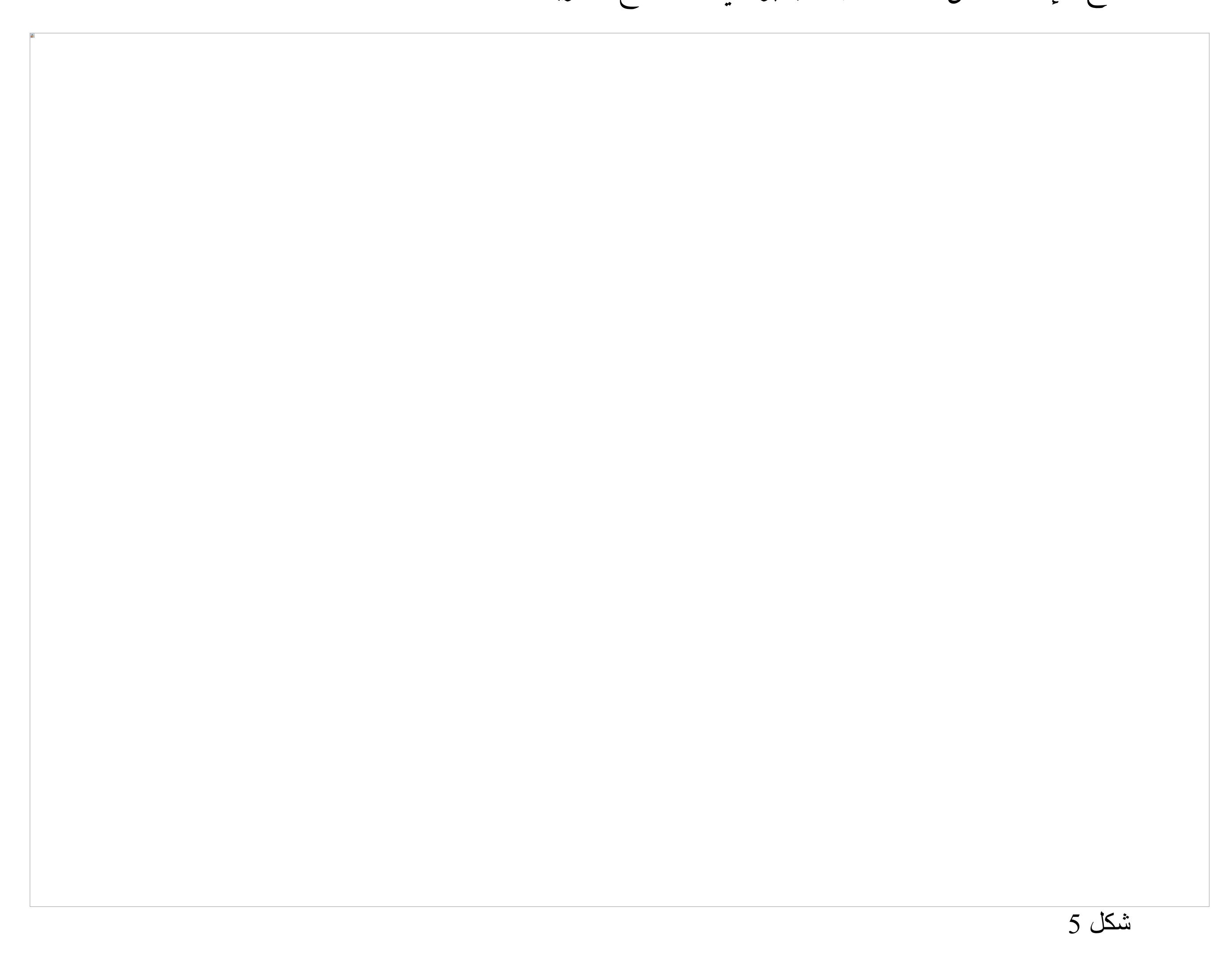
تسامح مع الأخطاء في أحد الأقراص، آخذين بالاعتبار أنه (في حال قرصين على سبيل المثال) المعلومات نفسها موجودة. عادة ما يكون RAID 1 ملائماً للأنظمة ذات التواجد الدائم، كالبيئات المتاحة على مدار الساعة وكلّ أيام الأسبوع (24×7)، والتي نكون فيها بحاجة ماسة إلى الموارد. يجعل هذا الإعداد من الممكن أيضاً(إذا كان العتاد يدعم ذلك) باستبدال الأقراص أثناء عمل النظام. فإذا اكتشفنا خطأً في أحد الأقراص، فيمكننا إزالة ذلك القرص ووضع آخر مكانه دون إيقاف النظام.



شكل 4

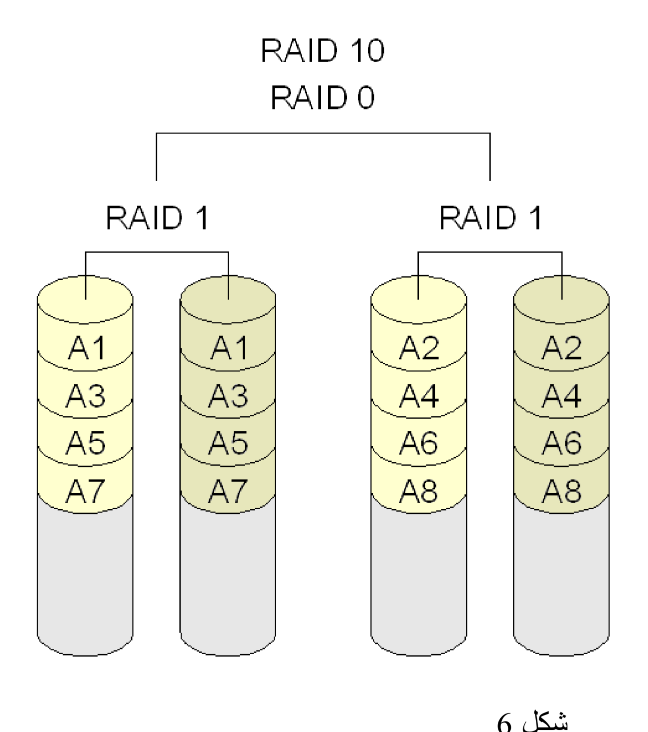
- RAID 2 : في الأنظمة سابقة الذكر، تُقسّم البيانات إلى كُتَل للتوزيع المتتالي؛ أما هنا، فالبيانات مقسمة إلى أجزاء ثُنائيّة (بِت bit)، وتُستخدم أكواد مكررة لتصحيح البيانات [ويطلق على هذه الأكواد parity]. هذا النوع غير مستخدم بكثرة، رغم المستوى العالي من الأداء الذي يمكن أن يوفره، حيث يحتاج افراضيًّا إلى عدد كبير من الأقراص ليعمل، واحد لكل بِت من البيانات، والعديد منها لحساب التكرار (فمثلاً في أنظمة 32بت، سيكون علينا استخدام 39 قرص).

- RAID 3 : يستخدم تقسيمة تعتمد على الثُمانيّات (bytes)، وقرص لكُتَل التصحيح (parity blocks). وهذا النوع غير مستخدم بكثرة أيضاً، حيث أنها اعتماداً على حجم البيانات وتوزيعها فهي لا توفر وصولاً متزامناً. RAID 4 مشابه له أيضاً، لكنه يقسِّم البيانات على مستوى الكُتل، بدلاً من التقسيم على مستوى الوحدة الثُمانيّة (بايت)، مما يعني أنه من الممكن تقديم طلبات متزامنة بينما تُطلَب كتلة واحدة.
- RAID 5 : يُستخدَم التقسيم على مستوى الكُتَل، وتوزّع بيانات التصحيح على الأقراص. يُستخدَم بكثرة نظراً لبساطة نظام التصحيح ولإمكانية عمل هذه الحسابات بسهولة في العتاد، مع مستويات أداء جيدة.



شكل 5

- RAID 0+1 (أو 01) : تقسيم ومرآة، وهو مستوى مركَّب؛ فمثلاً، ننشئ مجموعتي RAID 0 ثم نستخدمهما في RAID 1 لإنشاء مرآة بينهما. من فوائده أنه في حال حدوث خطأ، يمكن إعادة بناء RAID 0 بفضل النسخة الأخرى، لكن إذا احتجنا لإضافة المزيد من الأقراص، فيجب علينا إضافتها إلى كل مجموعات RAID 0 بالتساوي.
- RAID 10 (أو RAID 1+0) : تقسيم المرايا، وهي مجموعات من RAID 1 تحت RAID 0. بهذا الطريقة، يمكن أن يفشل قرص في كلّ مجموعة RAID 1 دون التسبب بخسارة البيانات. لكن بالطبع يعني هذا أننا سنحتاج لتغييرها، وإلا فسيصير القرص الآخر المتبقي في النظام سبباً آخر محتملاً للخطأ. يُستخدَم هذا الإعداد عادة في لقواعد البيانات عالية الأداء (بسبب التسامح مع الأخطاء والأداء العالي، حيث انها لا تعتمد على بيانات التصحيح).



شكل 6

هنا بعض النقاط التي علينا اخذها بعين الاعتبار فيما يخص RAID بشكل عام:

- يساعد RAID على إبقاء النظام يعمل لمدّة أطول، حيث تجعل بعض المستويات من الممكن أن يستمر النظام بالعمل بشكل سليم عند فشل أقراص، ويمكن أيضاً – اعتماداً على العتاد المستخدَم – تبديل العتاد الذي به مشكلة أثناء عمل النظام دون الحاجة لإيقافه، وهو أمر هام جدًّا في الأنظمة الحساسة.
- يمكن لريد أن يحسِّن أداء التطبيقات، خاصة في الأنظمة التي فيها مرآة، حيث يسمح تقسيم البيانات لعمليات القراءة المتتالية بأن تحسن بشكل ملحوظ، حيث يمكن للأقراص أن تقدِّم إمكانية القراءة المتزامنة، مما يزيد من

سرعة نقل البيانات.

- لا يحمي RAID البيانات، من البديهي أنه لا يحمي البيانات من المشاكل الأخرى الممكنة (كالفيروسات، أو الأخطاء العامة، أو الكوارث الطبيعية). علينا أن نعتمد على أنماط النسخ الاحتياطي.
- استعادة البيانات ليست سهلة. إذا كان القرص منتمياً إلى مصفوفة ريد، يفترض أن تتم الاستعادة منه في تلك البيئة. من الضروري وجود برمجية خاصة بمشغل العتاد للوصول إلى البيانات.
- ومن ناحية أخرى، فهو في العادة لا يزيد أداء تطبيقات المستخدم العادية، حتى وإن كانت تطبيقات سطح مكتب، لأن هذه التطبيقات لها مكونات تتصل بالذاكرة RAM وقليل من البيانات، مما يعني أنها لن تستفيد من القراءة المتتالية أو النقل الدائم للبيانات. في هذه البيئات، يمكن أن يكون التحسن المحتمل في الأداء والكفاءة غير ملحوظ حتّى.
- نقل المعلومات لا يتحسّن أو يسهل بأي طريقة؛ دون ريد، من السهل نقل البيانات، وذلك ببساطة بنقل القرص من نظام إلى آخر. في حالة RAID، من المستحيل تقريباً (مالم يكن لدينا نفس العتاد) نقل مصفوفة واحدة من الأقراص إلى نظام آخر.

في جنو/لينكس، RAID مدعوم عبر العديد من وحدات النواة المرتبطة بمجموعات مختلفة من المصنّعين أو الرقاقات لمشغلات RAID هذه. يسمح هذا للنظام بأن يعزل نفسه عن آليات العتاد، ويجعلها غير مرئية للنظام وللمستخدم النهائي. وعلى أي حال، تسمح لنا وحدات النواة هذه بالوصول إلى تفاصيل هذه المشغلات وضبط معاملاتها على مستوى منخفض جدًّا، والتي يمكن في بعض الحالات (خاصة في الخوادم التي تدعم حمل دخل/خرج عالٍ) أن تكون مفيدة لتضبيط نظام الأقراص الذي يستخدمه الخادم للوصول إلى أعلى أداء ممكن للنظام.

الخيار الآخر الذي سنحلّله هو توصيل هذه العمليات عبر مكونات برمجية، وتحديداً المكونات البرمجية لدعم RAID في نظام جنو/لينكس.

لنواة جنو/لينكس وحدة من النوع المسمّى Multiple Device – md والتي يمكننا اعتبارها دعماً من مشغّل النواة لـ RAID. يمكننا عبر هذا المشغل تنفيذ RAID بالمستويات 0،1،4،5 ومستويات RAID المركّبة (مثل RAID 10) على أجهزة كل مختلفة مثل أقراص IDE و SCSI. هناك أيضاً المستوى الخطّي، والذي يجمع الأقراص المتاحة بشكل خطيّ (ولا يهم إذا كانت بأحجام مختلفة)، مما يعني بأن الأقراص يكتب عليها بالتتابع.

لاستخدام برمجيات RAID في لينكس، يجب أن يكون دعم RAID متوفراً لدينا في النواة، وأن تكون وحدة md مفعّلة – في الحالات المعتمدة عليها –، إضافة إلى بعض المشغلات اعتماداً على الحالة (انظر الى المشغلات المتاحة المرتبطة بـ RAID، مثل modconf في دبيان). الطريقة المفضلة لتنفيذ مصفوفات RAID عبر برمجية RAID يوفرها لينكس، يتم إما أثناء التثبيت، أو عبر أداة mdadm. تسمح لنا هذه الأداة بإنشاء وإدارة هذه المصفوفات.

لنلقِ نظرة على بعض الأمثلة (سنفترض أننا نعمل على أقراص SCSI، مثل dev/sda/، و dev/sdb/ ....والتي لدينا فيها العديد من الأقسام المتاحة لتنفيذ RAID عليها):

إنشاء مصفوفة خطية:

```
# mdadm --create --verbose /dev/md0 --level=linear --raid-devices=2 /dev/sda1 /dev/sda2
```

وبهذا ننشئ مصفوفة خطية معتمدة على الأقسام الأولى للقرصين sda و sdb، بإنشاء الجهاز dev/md0/، والذي يمكن استخدامه الآن كقرص جديد (على فرض أن نقطة الضم media/diskRAID/ موجودة):

```
# mkfs.ext2fs /dev/md0
# mount /dev/md0 /media/diskRAID
```

لعمل RAID 0 أو RAID 1، يمكننا ببساطة تغيير المستوى (level--) إلى RAID 0 أو RAID 1. يمكننا بالأمر mdadm --detail /dev/md0 أن نتفقد معاملات المصفوفة المُنشأة حديثاً. يمكننا أيضاً مراجعة معلومات المدخلة mdstat في /proc/ لتحديد المصفوفات الفعالة ومعاملاتها. يمكننا أيضاً في حالة المرايا أن نرى الإنشاء الابتدائيّ للنسخة الاحتياطية (كما في المستويات 1 و 5) في ذاك الملف؛ سنرى في proc/mdstat/ مستوى إعادة الإنشاء (والوقت المقدر لإكمال العملية).

توفر أداة mdadm العديد من الخيارات التي تسمح لنا بتجربة وإدارة مصفوفات RAID البرمجية المختلفة المُنشأة (يمكننا

أن نرى وصفاً وأمثلة في man mdadm).

وهناك أمر آخر هام للأخذ بعين الاعتبار، ألا وهو التحسينات التي يفترض أن يتم عملها لمصفوفات RAID وذلك لتحسين الأداء، وذلك عبر مراقبة سلوكه لتحسين معاملات نظام الملفات، إضافة إلى استخدام مستويات RAID ومزاياها بطريقة أكثر فاعلية.

7.2 مدير الأقراص المنطقية LVM

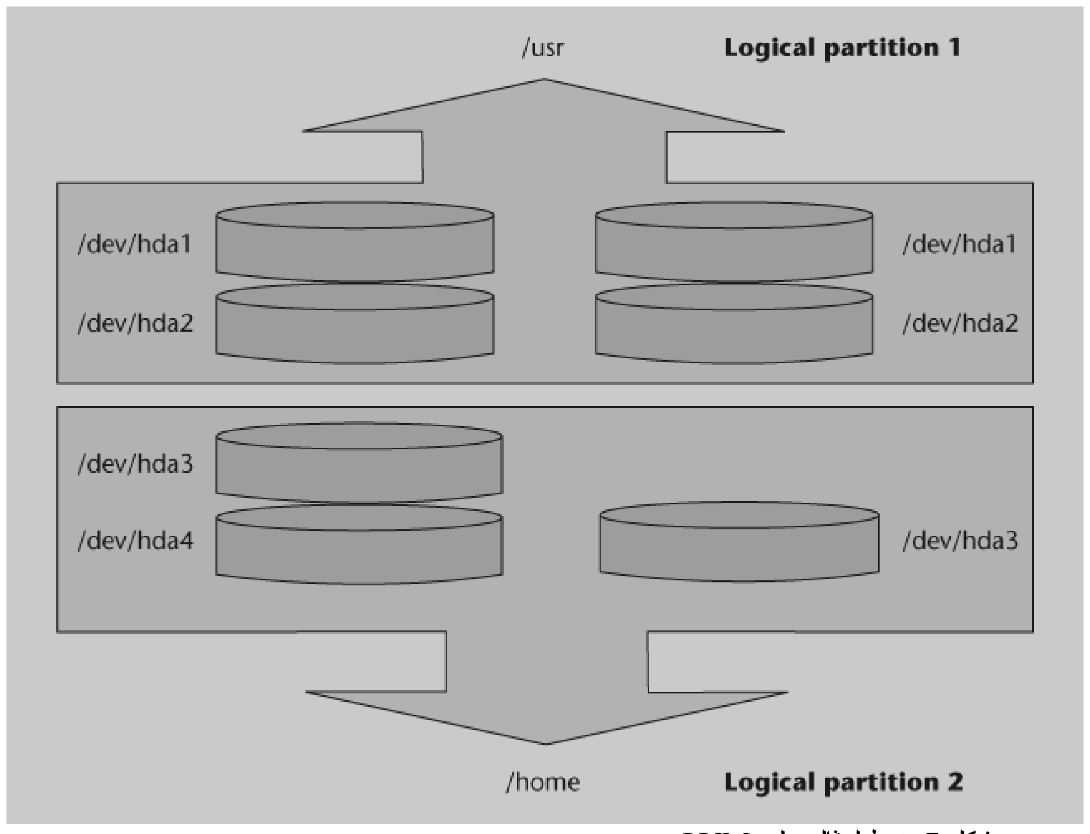
هناك حاجة للابتعاد عن النظام الأقراص الفيزيائية وإعداداتها وعدد الأجهزة، وبهذا يمكن لنظام التشغيل أن يقوم بهذا العمل وبهذا لا نعود بحاجة للاهتمام بهذه المعاملات مباشرة. وبهذا، يمكن أن نرى نظام إدارة الأقراص المنطقية كطبقة للتخزين الافتراضي يتيح مشهداً أبسط وأسلس في الاستخدام.

يوجد في لينكس مدير أقراص منطقية Logical Volume Manager – LVM مبني على فكرة طوّرت من مدراء وسائط التخزين المستخدمة في HP-UX (وهو نظام يونكس مملوك من HP). هناك الآن إصداران و LVM2 هو الأكثر استخداماً بسبب عدد من المزايا المضافة.

تتكون معمارية LVM بالعادة من العناصر (الرئيسية) التالية:

- **الوسائط الفيزيائيّة Physical Volumes – PV** : وهي أقراص صلبة أو أقسام من أقراص أو أي عنصر آخر يظهر كقرص صلب في النظام (مثل RAID البرمجي أو العتادي).
- **الوسائط المنطقية Logical Volumes – LV** : هذه تعادل الأقسام على القرص الفيزيائيّ. يظهر الوسط المنطقي LV كجهاز كُتَل خام (ويعادل تماماً قسماً فيزيائيًّا)، ويمكن أن يحوي نظام ملفات (كمجلد المنزل /home/ لأحد المستخدمين مثلاً) في العادة، تبدو الوسائط أكثر منطقية للمدراء، حيث يمكن استخدام أسماء لتحديدها (فمثلاً، يمكننا أن نستخدم جهازاً منطقياً اسمه stock أو marketing بدلاً منى hda6 أو sda3).
- **مجموعات الوسائط Volume Groups – VG** : هذا العنصر في الطبقة العليا. وهو الوحدة الإدارية التي تحوي الموارد التي لدينا، سواء كانت وسائط منطقية LV، أو فيزيائيّة PV. البيانات المتاحة في الوسائط الفيزيائيّة، وطريقة تكوين الوسائط المنطقية باستخدام الوسائط الفيزيائية محفوظة في هذه الوحدة. ومن الواضح أنه يجب أن يكون لدينا وسائط فيزيائيّة مصنفة في وحدات منطقية LV مختلفة ليكون عندنا مجموعة وسائط VG.

على سبيل المثال، يمكننا في الشكل التالي أن نرى مجموعات وسائط لدينا فيها 7 وسائط فيزيائية (على شكل أقسام لقرص صلب) مجمعة لتكوين وسيطين منطقيين (تم عملهما لتكوين نظامي الملفات /usr/ و /home/).



شكل 7: نمط لمثال على LVM

يمكننا باستخدام وسائط منطقية التعامل مع مساحة التخزين المتاحة (والتي يمكن أن تتكون من عدد كبير من الأقراص والأقسام المختلفة) بمرونة أكبر، بناء على الاحتياجات التي تظهر، ويمكننا أن ندير المساحة عبر محددات مناسبة أكثر وعبر عمليات تسمح لنا بتطويع المساحة لتناسب الاحتياجات التي تظهر في أي لحظة.

تسمح لنا إدارة الوسائط المنطقية بما يلي:

- إعادة تحجيم المجموعات والوسائط المنطقية، وذلك بإضافة وسائط فيزيائية جديدة، أو إزالة بعض الوسائط الموجودة.
- أخذ لقطات لنظام الملفات (قراءة فقط في LVM1، وقراءة و/أو كتابة في LVM2). هذا يجعل من الممكن إنشاء جهاز جديد يشكل صورة (أو لقطة) لحالة الوسيط المنطقي. ويمكننا كذلك إنشاء لقطة، وضمّها، وتجربة عدد من العمليات أو ضبط برمجيات جديدة أو عناصر أخرى، وإذا لم تعمل كما كنا نتوقع، فيمكننا إرجاع الوسط المنطقي الأصلي إلى الحالة التي كان عليها قبل القيام بهذه الاختبارات.
- RAID 0 على الوسائط المنطقية.

المستويات 1 و 5 من RAID غير مضمنة في LVM، فإذا كانت ضرورية (أو بعبارة أخرى، إذا كان التكرار والتسامح مع الأخطاء مطلوبين)، فعلينا عندها أن نستخدم مشغلات إما RAID البرمجي أو العتادي التي ستنفّذها وأن نستخدم LVM كطبقة فوقيّة عليها.

سنورد مثالاً اعتياديًّا مختصراً (وفي العديد من الحالات يحوي مثبِّت التوزيعة عملية مشابهة إذا استخدمنا LVM كنظام التخزين الابتدائيّ). وبشكل أساسيّ، يتوجّب علينا: 1) إنشاء وسائط فيزيائيّة PV. و 2) إنشاء المجموعة المنطقية VG ثم 3) إنشاء القرص المنطقي، وفي النهاية، إنشاء واستخدام نظام الملفات.

1) مثال: لدينا ثلاثة أقسام على أقراص منفصلة، وقد أنشأنا ثلاثة وسائط فيزيائيّة، وأنشأنا المحتوى:

```
# dd if=/dev/zero of=/dev/hda1 bs=1k count=1
# dd if=/dev/zero of=/dev/hda2 bs=1k count=1
# dd if=/dev/zero of=/dev/hdb1 bs=1k count=1
# pvcreate /dev/hda1
        Physical volume "/dev/sda1" successfully created
# pvcreate /dev/hda2
        Physical volume "/dev/sda2" successfully created
# pvcreate /dev/hdb1
        Physical volume "/dev/sdb1" successfully created
```

2) إنشاء مجموعة وسائط VG من الوسائط الفيزيائية المختلفة:

```
# vgcreate group_disks /dev/hda1 /dev/hda2 /dev/hdb1
        Volume group "group_disks" successfully created
```

3) ننشئ الوسيط المنطقي LV (وهو في هذه الحالة بسعة 1 جيجا)، بناء على العناصر التي لدينا في المجموعة VG (يحدد n- اسم الوسيط):

```
# lvcreate -L1G -n logical_volume group_disks
lvcreate – – doing automatic backup of "group_disks"
lvcreate – – doing logical volume "/dev/group_disks/ logical_volume" successfully created
```

وفي النهاية، نُنشئ نظام ملفات (وهو في هذه الحالة ReiserFS):

```
# mkfs.reiserfs /dev/group_disks/logical_volume
```

والذي يمكننا – مثلاً – أن نستخدمه كمساحة للنسخ الاحتياطيّ

```
# mkdir /mnt/backup
# mount -t reiserfs /dev/group_disks/logical_volume /mnt/backup
```

في النهاية، سيكون لدينا جهاز كوسيط منطقي لتنفيذ نظام ملفات في جهازنا.

8 تحديث البرمجيات

لتتمكن من إدارة التثبيت أو تحديث البرمجيات في نظامنا، سنعتمد – في البداية – على نوع حزم البرمجيات المستخدمة في نظامنا:

- RPM : حزم تستخدم توزيعات فيدورا/ردهات (ومشتقاتها). ويتم التعامل معها عادة بالأمر rpm. تحوي الاعتماديات التي تطلبها البرمجية من الحزم الأخرى. أو بمستوى أعلى، عبر yum (أو up2date في بعض التوزيعات المشتقة من ردهات).
- DEB : حزم دبيان التي يتم التعامل معها عادة عبر مجموعة من الأدوات التي تعمل بمستويات مختلفة مع حزم أو مجموعات منفردة. علينا أن نذكر منها: dselect, tasksel, dpkg, apt-get, ...
- TAR أو tgz (وهذه الأخيرة هي نفسها tar.gz): وهي ببساطة ملفات حزم تم جمعها وضغطها باستخدام أوامر قياسية مثل tar و gzip (وهي نفسها الأوامر المستخدمة لفك الضغط). هذه الحزم لا تحوي معلومات عن أيّ اعتماديات ويمكن بالعادة تثبيتها في أماكن مختلفة إذا لم تكن تحوي أيّ معلومات عن (مسار) جذر محدد.

هناك العديد من الأدوات الرسومية للتعامل مع هذه الحزم، مثل: Kpackage لحزم RPM، و Synaptic و Gnome-apt لحزم DEB، أما TGZ فعبر أداة Kpackage أو من مدير الملفات الرسومي نفسه (في جنوم وكدي). هناك أيضاً في العادة أدوات لتحويل الحزم. فعلى سبيل المثال، لدينا alien في دبيان، والذي يمكننا به تحويل حزم RPM إلى DEB. رغم أنه من الضروري أخذ الاحتياطات المناسبة – بحيث لا تغير الحزمة أي سلوك أو نظام بشكل غير متوقع – حيث أنها موجهة لتوزيعة أخرى.

اعتماداً على استخدام أنواع الحزم والادوات: سيكون من الممكن تثبيت أو تحديث البرمجية في نظامنا بطرق مختلفة:

1) من قرص التثبيت نفسه؛ في العادة، تبحث كل التوزيعات عن البرمجيات في القرص الضوئي. لكن يفترض أن يتم التحقق من البرمجية أنها ليست قديمة، وأنه لا يوجد ترقيع كتحديثات أو إصدارات جديدة ذات مزايا أكثر؛ وبهذا، إذا تم استخدام قرص للتثبيت، فمن الإجراءات المتبعة التحقق من أنها بأحدث إصدار وأنه لا توجد إصدارات

أحدث.

2) عبر التحديثات أو خدمات البحث عن برمجيات، سواء كانت مجانية – كما في apt-get في دبيان، و yum في فيدورا –، أو عبر خدمات الاشتراك (الخدمات المدفوعة أو الخدمات التي توفر الإمكانيات الأساسيّة)، مثل شبكة ردهات للإصدارات التجارية لردهات.

3) عبر مستودعات البرمجيات التي توفر حزم برمجيات مبنية مسبقاً لتوزيعة محدّدة.

4) من المُنشئ أو الموزّع الأصليّ للبرمجية الذي يمكن أن يوفر مجموعة من حزم تثبيت البرمجية. قد نجد أنفسنا غير قادرين على إيجاد حزم من النوع الذي نحتاجه لتوزيعتنا.

5) برمجيات غير محزمة، أو مع ضغط فقط، دون أي نوع من الاعتماديات.

6) مصدر برمجي فقط، على شكل حزمة أو ملف مضغوط.

9 الوظائف الدفعيّة

عادة ما يكون من الضروري في الأعمال الإدارية تنفيذ مهام معينة في أوقات زمنية معينة، إما لأنه من الضروري برمجة المهام بحيث تتم في الوقت الذي يكون فيه استخدام الجهاز أقل ما يمكن أو بسبب طبيعة تلك المهام التي يفترض أن تتم دوريًّا.

هناك العديد من الأنظمة التي تسمح لنا بعمل جدولة للمهام (تخطيط لتنفيذ مهام) لتنفيذ هذه المهام خارج وقت الدوام، كالخدمات الدورية أو المبرمجة التالية:

- nohup : هو بالأحرى أبسط أمر للمستخدمين بحيث يسمح لهم بتنفيذ أوامر غير تفاعلية فور خروجهم من الحساب. عادة، عندما يخرج مستخدم ما، فإنه يخسر عملياته؛ يسمح nohup للمستخدمين بترك العمليات تعمل حتى بعد خروج هذا المستخدم من النظام.
- at : يسمح لنا بتنفيذ مهام في وقت آخر، وذلك ببرمجة اللحظة المحددة التي نرغب بأن تُنفَّذ تلك المهمة فيها، وذلك بتحديد الوقت (hh:mm) والتاريخ، أو بتحديد ما إذا كانت ستنفذ اليوم أو غداً. مثال:

  at 10pm task

  لتنفيذ المهمة في الساعة العاشرة ليلاً.

  at 2am tomorrow task

  لتنفيذ المهمة في الساعة الثانية فجراً.
- cron : يسمح لنا بعمل قائمة بالمهام التي سيتم تنفيذها مع البرمجة المرتبطة بها؛ يتم حفظ هذه الإعدادات في /etc/crontab، وبالتحديد، في كل مدخلة في هذا الملف، لدينا: الوقت (الساعة والدقيقة) الذي سيتم فيه تنفيذ المهمة، ورقم اليوم في الشهر، ورقم الشهر، رقم اليوم في الأسبوع[9]، إضافة إلى العنصر الذي سيتم تنفيذه (والذي قد يكون مهمة أو مجلداً يحوي المهام التي ستُنفَّذ). على سبيل المثال، المحتوى القياسي يبدو مثل:

---

9 بحيث يبدأ الأسبوع بيوم الأحد الذي يأخذ الرقم 0، وينتهي بيوم السبت الذي يحمل الرقم 6.

```
25 6 * * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily
47 6 * * 7 root test -e /usr/sbin/anacron || run-parts --report /etc/cron.weekly
52 6 1 * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.monthl
```

والذي تبرمج فيه مجموعة من الأوامر للتنفيذ: يومياً (* تعني "في أي يوم")، أسبوعياً (في اليوم السابع من الأسبوع)، أو شهرياً (في اليوم الأول من كل شهر). عادة ما يتم تنفيذ المهام بالأمر crontab، لكن نظام cron يعتبر أن الجهاز دائماً يعمل، وإن لم يكن كذلك، فمن الأفضل استخدام anacron، والذي يتحقق مما إذا كانت المهمة قد نُفِّذَت عندما كان يفترض أن يتم ذلك أم لا، وإن لم تكن قد نُفِّذَت، فسيتم تنفيذ المهمة.

يتم تفقد كل سطر من الملف السابق للتأكد من أن الأمر anacron موجود، وأن النصوص البرمجية المرتبطة بكل حدث قد تم تنفيذها، في هذه الحالة، يتم حفظها في مجلدات معدّة لهذا الغرض.

قد يكون هناك أيضاً cron.allow و cron.deny لتحديد من يمكنه أو لا يمكنه وضع هذه المهام في cron. يمكن للمستخدم – عبر الأمر crontab – تعريف مهمة بنفس الطريقة التي رأيناها سابقاً، والتي تكون في العادة محفوظة في /var/spool/cron/crontabs. وفي بعض الحالات أيضاً يكون هناك مجلد /etc/cron.d/، حيث يمكننا وضع المهام ليتم التعامل معها كما لو كانت امتدادات (أو إضافات) إلى ملف /etc/crontab.

10 درس تعليمي: مهام مجمعة من الفصول المختلفة

سنبدأ بتفحص الحالة العامة لنظامنا. سننفذ هذه الخطوات على نظام دبيان. هذا نظام دبيان غير مستقر (الإصدار غير المستقر، لكنه محدّث أكثر)؛ لكن الإجراءات المختلفة – غالباً – يمكن تنفيذها على التوزيعات المختلفة، مثل فيدورا/ردهات (سنذكر بعض أكثر التغييرات أهمية). يتكون العتاد من جهاز Pentium 4 بسرعة معالج 2.66 GHz وذاكرة RAM قدرها 768 MB، وعدد من الأقراص، ناسخ أقراص CD و DVD، إضافة إلى طرفيات أخرى سنجمع معلومات عنها بينما نتقدم خطوة بخطوة.

أولاً، سنرى كيف أقلع نظامنا في آخر مرة:

```
# uptime
17:38:22 up 2:46, 5 users, load average: 0.05, 0.03, 0.04
```

يخبرنا هذا الأمر بالمدة التي بقي فيها النظام يعمل منذ آخر إقلاع له – وهو ساعتان وستة وأربعون دقيقة –، ولدينا في هذه الحالة خمسة مستخدمين. قد لا يشير هذا بالضرورة إلى خمسة مستخدمين مختلفين، ولكنها تكون بالعادة جلسات مستخدمين مفتوحة (من طرفية واحدة على سبيل المثال). يعرض الأمر who قائمة بهؤلاء المستخدمين. متوسط الحمل هو متوسط حمل النظام في آخر دقيقة وآخر خمس دقائق وآخر خمس عشرة دقيقة.

لنلقِ نظرة على سجل إقلاع النظام (بالأمر dmesg)، والسطور التي نشأت عندما أقلع النظام (لقد حذفنا بعض السطور بهدف التوضيح):

```
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org)
(gcc version 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr
15 21:03:57 UTC 2007
BIOS-provided physical RAM map:
BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
```

```
BIOS-e820: 00000000000ce000 - 00000000000d0000 (reserved)
BIOS-e820: 00000000000dc000 - 0000000000100000 (reserved)
BIOS-e820: 0000000000100000 - 000000002f6e0000 (usable)
BIOS-e820: 000000002f6e0000 - 000000002f6f0000 (ACPI data)
BIOS-e820: 000000002f6f0000 - 000000002f700000 (ACPI NVS)
BIOS-e820: 000000002f700000 - 000000002f780000 (usable)
BIOS-e820: 000000002f780000 - 0000000030000000 (reserved)
BIOS-e820: 00000000ff800000 - 00000000ffc00000 (reserved)
BIOS-e820: 00000000fffffc00 - 0000000100000000 (reserved)
0MB HIGHMEM available.
759MB LOWMEM available.
```

هذه السطور الأولى تعطي بيانات مثيرة للاهتمام: إصدار نواة لينكس هو 2.6.20-1-686، أي الإصدار 2.6 المراجعة 20 على المراجعة الأولى لدبيان والمعدّة لأجهزة 686 (وهي معماريات إنتل x86 ذات 32بت). وتشير أيضاً إلى أننا نقلع نظام دبيان بهذه النواة التي بُنيت بمصرّف GNU gcc ذي الإصدار 4.1.2 وبالتاريخ المذكور. تلي ذلك خريطة مناطق الذاكرة المستخدمة (المحجوزة) من نظام الدخل/الخرج القياسي BIOS، تليها الذاكرة الكلية المكتشفة في الجهاز، وهي 759 ميجا، والتي يمكننا أن نضيف إليها أول 1 ميجا لنخرج بمحصلة قدرها 760 ميجا بايت.

```
Kernel command line: BOOT_IMAGE=LinuxNEW ro root=302 lang=es
acpi=force
Initializing CPU#0
Console: colour dummy device 80x25
```

```
Memory: 766132k/777728k available (1641k kernel code, 10968k reserved,

619k data, 208k init, 0k highmem)

Calibrating delay using timer specific routine.. 5320.63 BogoMIPS

(lpj=10641275)
```

هذا يخبرنا كيف أقلع الجهاز، وأي سطور أوامر تم تمريرها إلى النواة (العديد من الخيارات يمكن أن تمرّر، مثل lilo أو grub). ونحن نقلع في الطور النصيّ ذي 80 × 25 محرفاً (يمكن تغيير هذا). أما BogoMIPS فهي قياسات داخلية للنواة لسرعة المعالج. هناك معماريات يصعب فيها معرفة مقدار الـ MHz التي يعمل بها المعالج، ولهذا يتم استخدام هذه الطريقة في قياس السرعة. ونتيجة لهذا، يتم إعطاؤنا بيانات أكثر عن الذاكرة الرئيسية ولم هي مستخدمة في هذه المرحلة من الإقلاع.

```
CPU: Trace cache: 12K uops, L1 D cache: 8K
CPU: L2 cache: 512K
CPU: Hyper-Threading is disabled
Intel machine check architecture supported.
Intel machine check reporting enabled on CPU#0.
CPU0: Intel P4/Xeon Extended MCE MSRs (12) available
CPU0: Intel(R) Pentium(R) 4 CPU 2.66GHz stepping 09
```

ويعطينا كذلك معلومات عديدة عن المعالج، ومنها: سعة الذاكرة cache ذات المستوى الأول، وسعة cache الداخلية للمعالج، و L1 المقسمة على TraceCache في Pentium 4 (أو على تعليمات cache)، والذاكرة المخبأة للبيانات، وذاكرة المستوى الثاني المشتركة L2 cache، ونوع المعالج وسرعته، ومنفذ النظام المرتبط به.

```
PCI: PCI BIOS revision 2.10 entry at 0xfd994, last bus=3
Setting up standard PCI resources
...
NET: Registered protocol
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
checking if image is initramfs... it is
Freeing initrd memory: 1270k freed
fb0: VESA VGA frame buffer device
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing enabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
00:09: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
RAMDISK driver initialized: 16 RAM disks of 8192K size 1024 blocksize
PNP: PS/2 Controller [PNP0303:KBC0,PNP0f13:MSE0] at 0x60,0x64 irq 1,12
i8042.c: Detected active multiplexing controller, rev 1.1.
serial: i8042 KBD port at 0x60,0x64 irq 1
serial: i8042 AUX0 port at 0x60,0x64 irq 12
serial: i8042 AUX1 port at 0x60,0x64 irq 12
serial: i8042 AUX2 port at 0x60,0x64 irq 12
serial: i8042 AUX3 port at 0x60,0x64 irq 12
mice: PS/2 mouse device common for all mice
```

تستمر النواة والأجهزة بالإقلاع، ذاكرة بدء موافيق الشبكة. الطرفيات، والمنافذ التسلسلية ttyS0 (والتي قد تكون com1 و ttyS1 (أي com2). وتوفر معلومات عن الذاكرة RAM المستخدمة، واكتشاف أجهزة PS2، كالفأرة ولوحة المفاتيح.

```
ICH4: IDE controller at PCI slot 0000:00:1f.1

ide0: BM-DMA at 0x1860-0x1867, BIOS settings: hda:DMA, hdb:pio
ide1: BM-DMA at 0x1868-0x186f, BIOS settings: hdc:DMA, hdd:pio
Probing IDE interface ide0...
hda: FUJITSU MHT2030AT, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
Probing IDE interface ide1...
hdc: SAMSUNG CDRW/DVD SN-324F, ATAPI CD/DVD-ROM drive
ide1 at 0x170-0x177,0x376 on irq 15
SCSI subsystem initialized
libata version 2.00 loaded.
hda: max request size: 128KiB
hda: 58605120 sectors (30005 MB) w/2048KiB Cache, CHS=58140/16/63<6>hda:
hw_config=600b
, UDMA(100)
hda: cache flushes supported
hda: hda1 hda2 hda3
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted file system with ordered data mode.
hdc: ATAPI 24X DVD-ROM CD-R/RW drive, 2048kB Cache, UDMA(33)
Uniform CD-ROM driver Revision: 3.20
Addinf 618492 swap on /dev/hda3.
```

اكتشاف أجهزة IDE، اكتشاف شرائح IDE في منفذ PCI و الإخبار عمّا يستخدم الجهاز: hda و hdc، وهي [في هذه الحالة] على الترتيب: قرص صلب (fujitsu)، ومشغل DVD وناسخ أقراص CD من Samsung (على اعتبار أننا في هذه الحالة لدينا وحدة Combo[10]). وتشير هذه البيانات إلى وجود أقسام تعمل من القرص الصلب. يلي هذا اكتشاف النواة لنظام الملفات الرئيسي للينكس – وهو النظام السجليّ ext3 – الذي يفعِّل ويضيف مساحة الإبدال swap المتاحة أحد الأقسام.

10 تستخدم كلمة combo للإشارة إلى مشغلات الأقراص الضوئية التي يمكنها القراءة والكتابة على أقراص CD والقراءة من أقراص DVD دون إمكانية الكتابة عليها. ويطلق هذا الاسم أيضاً على المشغلات التي يمكنها قراءة أقراص BlueRay دون الكتابة إليها، مع إمكانيات الكتابة على أقراص CD و DVD.

```
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
input: PC Speaker as /class/input/input1
USB Universal Host Controller Interface driver v3.0
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
uhci_hcd 0000:00:1d.1: UHCI Host Controller
uhci_hcd 0000:00:1d.1: new USB bus registered, assigned bus number 2
uhci_hcd 0000:00:1d.1: irq 11, io base 0x00001820
usb usb2: configuration #1 chosen from 1 choice
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
hub 4-0:1.0: USB hub found
hub 4-0:1.0: 6 ports detected
```

اكتشاف للمزيد من الأجهزة، مثل USB (والوحدات المرتبطة)؛ وهي في هذه الحالة، موزّعان اثنان (بمجموع قدره 8 منافذ USB) تم اكتشافها.

```
parport: PnPBIOS parport detected.
parport0: PC-style at 0x378 (0x778), irq 7, dma 1
[PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
input: ImPS/2 Logitech Wheel Mouse as /class/input/input2
ieee1394: Initialized config rom entry 'ip1394'
eepro100.c:v1.09j-t 9/29/99 Donald Becker
Synaptics Touchpad, model: 1, fw: 5.9, id: 0x2e6eb1, caps: 0x944713/0xc0000
input: SynPS/2 Synaptics TouchPad as /class/input/input3

agpgart: Detected an Intel 845G Chipset
agpgart: Detected 8060K stolen Memory
agpgart: AGP aperture is 128M
eth0: OEM i82557/i82558 10/100 Ethernet, 00:00:F0:84:D3:A9, IRQ 11.
Board assembly 000000-000, Physical connectors present: RJ45
e100: Intel(R) PRO/100 Network Driver, 3.5.17-k2-NAPI
usbcore: registered new interface driver usbkbd
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.

lp0: using parport0 (interrupt-driven).
ppdev: user-space parallel port driver
```

والاكتشاف النهائي لبقية الأجهزة: المنفذ التسلسلي، ونموذج الفأرة، ومنفذ FireWire (المسمى أيضاً IEEE1394)، وبطاقة الشبكة (من إنتل)، وشاشة اللمس، وبطاقة الرسوميات i845 – VGA. ومعلومات أخرى عن بطاقة الشبكة، وهي intel pro 100، وتسجيل USB كوحدة تخزين (مما يشير لوجود جهاز تخزين متصل عبر USB كقرص خارجي[11])، واكتشاف المنافذ التسلسلية.

يمكننا أيضاً أن نرى كل هذه المعلومات – التي وصلنا إليها بالأمر dmesg – مخزنة في سجل النظام الرئيسي، وهو /var/log/messages. يمكننا في هذا السجل إيجاد رسائل النواة – إضافة إلى غيرها –، ورسائل المراقبات وأخطاء الأجهزة والشبكة، والتي ترسل رسائلها إلى مراقب خاص اسمه syslogd، وهو المسؤول عن كتابة الرسائل في هذا الملف. إذا كنا أقلعنا الجهاز مؤخراً، فسنرى أن آخر سطور تحوي نفس المعلومات التي يخرجها الأمر dmesg تماماً، فعلى سبيل المثال، إذا نظرنا إلى

11 قد يكون قرص فلاش أو ذاكرة كامرا أو قرصاً صلباً خارجياً ...إلخ.

الجزء الأخير من الملف (والذي عادة ما يكون كبيراً جدًّا):

```
# tail 200 /var/log/messages
```

فيمكننا أن نرى نفس السطور السابقة وبعض المعلومات الإضافية مثل:

```
shutdown[13325]: shutting down for system reboot
kernel: usb 4-1: USB disconnect, address 3
kernel: nfsd: last server has exited
kernel: nfsd: unexporting all file systems
kernel: Kernel logging (proc) stopped.
kernel: Kernel log daemon terminating.

exiting on signal 15
syslogd 1.4.1#20: restart.

kernel: klogd 1.4.1#20, log source = /proc/kmsg started.
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org) (gcc version 4.1.2
20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr 15 21:03:57 UTC 2007
kernel: BIOS-provided physical RAM map:
```

يتعلق القسم الأول بإيقاف التشغيل السابق للنظام، ويخبرنا بأن النواة توقفت عن وضع معلومات في /proc/، وأن النظام ينطفئ ... في بداية الإقلاع التالي، يتم تفعيل المراقب Syslogd الذي يُنشئ السجلات، ويبدأ النظام بالتحميل، مما يخبرنا بأن النواة ستبدأ بكتابة المعلومات في نظامها، ألا وهو /proc/؛ نحن ننظر إلى السطور الأولى من dmesg والتي تظهر إصدار النواة التي يتم تحميلها ومن ثم نجد ما رأينا من مخرجات dmesg.

في هذه المرحلة، هناك أمر آخر مفيد لمعرفة كيف تمت عملية الإقلاع، ألا وهو lsmod، والذي يخبرنا بالوحدات التي تم تحميلها في النواة (نسخة مختصرة):

```
# lsmod Module Size Used by
nfs 219468              0
nfsd 202192             17
exportfs 5632           1 nfsd
lockd 58216             3 nfs,nfsd
nfs_acl 3616            2 nfs,nfsd
sunrpc 148380           3 nfs,nfsd,lockd,nfs_acl
ppdev 8740              0
lp 11044                0
button 7856             0
ac 5220                 0
battery 9924            0
md_mod 71860            1
dm_snapshot 16580       0
dm_mirror 20340         0
dm_mod 52812            2 dm_snapshot,dm_mirror
i810fb 30268            0
vgastate 8512           1 i810fb
eeprom 7184             0
thermal 13928           0
processor 30536         1 thermal
fan 4772                0
udf 75876               0
ntfs 205364             0
usb_storage 75552       0
hid 22784               0
usbkbd 6752             0
eth1394 18468           0
e100 32648              0
eepro100 30096          0
ohci1394 32656          0
ieee1394 89208          2 eth1394,ohci1394
snd_intel8x0 31420      1
snd_ac97_codec 89412        1 snd_intel8x0
ac97_bus 2432           1 snd_ac97_codec
parport_pc 32772        1
snd 48196               6 snd_intel8x0,snd_ac97_codec,snd_pcm,snd_timer
ehci_hcd 29132          0
ide_cd 36672            0
cdrom 32960             1 ide_cd
```

```
soundcore 7616        1 snd
psmouse 35208         0
uhci_hcd 22160        0
parport 33672         3 ppdev,lp,parport_pc
intelfb 34596         0
serio_raw 6724        0
pcspkr 3264           0
pci_hotplug 29312     1 shpchp
usbcore 122312        6 dvb_usb,usb_storage,usbkbd,ehci_hcd,uhci_hcd
intel_agp 22748       1
agpgart 30504         5 i810fb,drm,intelfb,intel_agp
ext3 121032           1
jbd 55368             1 ext3
ide_disk 15744        3
ata_generic 7876      0
ata_piix 15044        0
libata 100052         2 ata_generic,ata_piix
scsi_mod 133100       2 usb_storage,libata
generic 4932          0 [permanent]
piix 9540             0 [permanent]
ide_core 114728       5 usb_storage,ide_cd,ide_disk,generic,piix
```

نجد بأنه لدينا بالأساس مشغلات العتاد الذي تم اكتشافه وعناصر أخرى ذات علاقة أو عناصر نحتاجها كاعتماديات.

ومن ثم، فإن هذا يعطينا لمحة عن الكيفية التي تم بها تحميل النواة ووحداتها. في هذه العملية، قد نكون رأينا خطأً إذا لم يكن العتاد قد ضُبِطَ بشكل مناسب، أو إذا كانت هناك وحدات للنواة لم تُصرَّف بشكل مناسب (لم تصرّف لإصدار النواة المناسب) أو غير موجودة، إلخ.

الخطوة التالية هي تفحّص العمليات في النظام، مثل أمر ps (وهي اختصار لعبارة "حالة العمليات" process status)، فمثلاً (يظهر لنا عمليات النظام وليس عمليات المستخدمين):

```
# ps -ef
UID     PID     PPID   C        STIME          TTY    TIME   CMD
```

معلومات العملية، حيث UID المستخدم الذي نفّذ العملية (أو المعرف الذي نُفّذَت به)، و PID كود العملية المُعطى من النظام يظهر في هذا العمود تصاعدياً، بالترتيب الذي بدأت فيه هذه العمليات [من الأقدم إلى الأحدث]؛ الأولى دائماً يكون رقمها صفراً، وتكون مرتبطة بالعملية init. أما PPID فهو رقم ID للعملية الأم. و STIME وهو الوقت الذي بدأت به العملية، و TTY الطرفية المرتبطة بالعملية (إذا كان هنالك واحدة)، و CMD وهو الأمر الذي نُفِّذَت به.

```
root 1 0 0 14:52 ? 00:00:00 init [2]
root 3 1 0 14:52 ? 00:00:00 [ksoftirqd/0]
root 143 6 0 14:52 ? 00:00:00 [bdflush]
root 145 6 0 14:52 ? 00:00:00 [kswapd0]
root 357 6 0 14:52 ? 00:00:01 [kjournald]
root 477 1 0 14:52 ? 00:00:00 udevd --daemon
root 719 6 0 14:52 ? 00:00:00 [khubd]
```

العديد من مراقبات النظام، مثل مراقب kswapd الذي يتحكم بإبدال الذاكرة الافتراضية. التعامل مع فيض النظام (bdflush). التعامل مع سجلات نظام الملفات (kjournald)، والتعامل مع USB (عبر khubd). أو مراقب udev الذي يتحكم بالتوصيل الفوري للأجهزة. وبشكل عام، فالمراقبات لا تعرف دائماً بوجود d في نهاية اسمها، وإذا كان لها k في البداية، فغالباً تكون خيوطاً داخلية للنواة.

```
root      1567 1 0 14:52 ? 00:00:00 dhclient -e -pf ...
root      1653 1 0 14:52 ? 00:00:00 /sbin/portmap
root      1829 1 0 14:52 ? 00:00:00 /sbin/syslogd
root      1839 1 0 14:52 ? 00:00:00 /sbin/klogd -x
root      1983 1 0 14:52 ? 00:00:09 /usr/sbin/cupsd
root      2178 1 0 14:53 ? 00:00:00 /usr/sbin/inetd
```

لدينا [هنا] dhclient، مما يشير إلى أن الجهاز عميل لخادم DHCP، وذلك للحصول على عنوان IP له. Syslogs، مراقب يرسل رسائل للسجلات. مراقب cups – كما ذكرنا سابقاً – مرتبط بنظام الطباعة. و inetd – كما سنرى في الجزء المتعلق بالشبكات – نوع من "superserver"، أو وسيط للمراقبات الأخرى المرتبطة بخدمات الشبكة.

```
root         2154 1 0 14:53 ?         00:00:00 /usr/sbin/rpc.mountd
root         2241 1 0 14:53 ?         00:00:00 /usr/sbin/sshd
root          2257 1 0 14:53 ?        00:00:00 /usr/bin/xfs -daemon
root         2573 1 0 14:53 ?         00:00:00 /usr/sbin/atd
root         2580 1 0 14:53 ?         00:00:00 /usr/sbin/cron
root         2675 1 0 14:53 ?         00:00:00 /usr/sbin/apache
www-data     2684 2675 0 14:53 ?      00:00:00 /usr/sbin/apache
www-data     2685 2675 0 14:53 ?      00:00:00 /usr/sbin/apache
```

هناك أيضاً sshd، وهو خادم آمن للوصول عن بعد (إصدار محسّن يسمح بالخدمات المتوافقة مع telnet و FTP)[12]. و XFS هو خادم الخطوط (نوع محرفيّ) لنظام X Window. الأوامر atd و cron يمكن استخدامها للتعامل مع المهام المبرمجة في وقت محدّد. أباتشي Apache خادم وِب، ويمكن أن يكون له عدد من الخيوط لاستقبال طلبات مختلفة.

```
root 2499 2493 0 14:53 ?       00:00:00 /usr/sbin/gdm
root 2502 2499 4 14:53 tty7    00:09:18 /usr/bin/X :0 -dpi 96 ...
root 2848 1 0 14:53 tty2       00:00:00 /sbin/getty 38400 tty2
root 2849 1 0 14:53 tty3       00:00:00 /sbin/getty 38400 tty3
root 3941 2847 0 14:57 tty1    00:00:00 -bash
root 16453 12970 0 18:10 pts/2 00:00:00 ps -ef
```

GDM هو الولوج الرسومي التابع لنظام المكتب جنوم (نقطة الدخول التي نُسأل فيها عن اسم الولوج وكلمة المرور)، وعمليات getty هي التي تدير الطرفيات النصية الافتراضية (والتي يمكننا أن نراها بالضغط على Alt+Fx[13] (أو Ctrl+Alt+Fx إذا كنّا في الطور الرسوميّ)). X هي عملية خادم الرسوميات X Window System وهي ضرورية لتنفيذ أية بيئة مكتب رسومية عليها. صدفة مفتوحة (bash)، و – في النهاية – العملية التي أنشأناها عندما طلبنا ps من سطر الأوامر.

---

12 لخادم ssh مزايا عديدة، وإمكانيات كبيرة، فهو يدعم طرقاً عديدة للاستيثاق منها المفتاح العام والخاص، والولوج دون كلمة مرور (عبر المفتاح)، والاتصال فيه يكون مشفراً، ويدعم نقل الملفات مباشرة عبر أدوات معدة لهذا الغرض مثل scp ويمكن استخدامه بتمرير الاتصال فيما يسمى ssh-tunneling ...إلخ. وهناك العديد من المواضيع في مجتمع لينكس العربي وفي مواقع أخرى عربية وأجنبية تشرح طرق استخدامه والاستفادة منه، منها مواضيع للدكتور علي الشمريّ حول ضبط الخادم وزيادة الأمن فيه، إضافة إلى الوثائق الرسمية للمشروع وللتوزيعات الكبيرة مثل فيدورا ودبيان وآرش وغيرها ...

13 تشير Fx هنا إلى أزرار الوظائف، بحيث X أحد الأرقام، F1, F2, F3 ...

يقدم الأمر ps عدداً من خيارات سطر الأوامر لضبط المعلومات التي نريدها عن كل عملية، سواء كان ذلك عن الوقت الذي بدأت فيه، أو نسبة استخدام المعالج، أو الذاكرة المستخدمة، إلخ. (ألقِ نظرة على دليل الاستخدام man لأداة ps). وههناك أمر آخر هام جداً وهو top، ويقوم بنفس وظيفة ps لكن بشكل ديناميكي؛ وبعبارة أخرى، تحدَّث كل قدر معين من الوقت، ويمكننا تصنيف العمليات باستخدام المعالج أو الذاكرة، ويقدم معلومات الحالة العامة للنظام.

من الأوامر الأخرى مفيدة لإدارة الموارد أمر free و vmstat، والذان يقدمان معلومات عن استخدام الذاكرة ونظام الذاكرة الافتراضية:

```
# free
        total        used        free    shared        buffers        cache
Mem:    767736       745232      22504    0            89564          457612
-/+ buffers/cache:   198056      569680
swap:   618492       1732        616760

# vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free    buff   cache   si  so  bi  bo  in   cs  us sy id wa st
 1  0   0      1732    22444  457640  0   0   71  28  508  614 12  3 83  2  0
```

الأمر free يظهر أيضاً مساحة الإبدال swap – والتي تساوي 600 ميجا تقريباً – وهي غير مستخدمة بشكل كبير في الوقت الراهن، حيث هناك مساحة ذاكرة فيزيائية كافية؛ فلدينا 22 ميجا حرة (وهذا يشير إلى استخدام كبير في الذاكرة الفيزيائية، والحاجة لاستخدام مساحة الإبدال قريباً). مساحتا الذاكرة والإبدال (كما في النواة 2.4) تضاف كل منهما إلى الأخرى لتكوين الذاكرة الكلية للنظام، وهي – في هذه الحالة – تعني وجود حصيلة قدرها 1.4 جيجا من الذاكرة المتاحة. قد يبدو [هذا القدر من الذاكرة] كبيراً، ولكن هذا يعتمد على التطبيقات التي يتم تشغيلها.

الأنشطة

1) تجعل مساحة الإبدال بالإمكان الإضافة إلى الذاكرة الفيزيائية وبهذا يكون لدينا ذاكرة افتراضية أكبر. اعتماداً على مقدار إضافة مساحة إضافية إلى الذاكرة الفيزيائية ومساحة الإبدال، هل يمكن أن تُستخدَم الذاكرة بالكامل؟ هل يمكننا حل هذه المعضلة بأي طريقة أخرى لا تعتمد على إضافة المزيد من الذاكرة الفيزيائية؟

2) لنفترض بأن لدينا نظام بقسمي لينكس: أحدهما / والآخر مساحة إبدال. كيف نحلّ المشكلة إذا استخدم حساب أحد المستخدمين كل مساحة القرص؟ وإذا كان لدينا قسم /home/ منفصل شارف أيضاً على الانتهاء، فكيف نحل هذه المشكلة؟

3) ثبّت نظام الطباعة CUPS، ثم عرف طابعتنا بحيث تعمل مع CUPS، ثم حاول إدارتها عبر واجهة الوِب. كما في حالة النظام الآن، هل يُنصَح بتغيير إعدادات CUPS المبدئيّة بأيّ طريقة؟ لماذا؟

4) اختبر الإعدادات المبدئية التي تأتي مع نظام جنو/لينكس للعمل غير التفاعلي باستخدام cron. أيّ الوظائف موجودة؟ ومتى تُنفّذ؟ هل هناك أفكار لأيّة وظائف جديدة يجب إضافتها؟

5) أعد صياغة تحليل لورشة العمل – أو الدرس التعليمي – (إضافة إلى أجزاء أخرى من هذه الوحدة) على الجهاز المتاح لديك. هل يمكننا إيجاد أيّة أخطاء أو أوضاع غير عادية في النظام الذي نختبره؟ إذا كان كذلك، فكيف نصححها؟

المراجع

مصادر أخرى للمراجع والمعلومات:

[Smi02] [Fri02] [Wm02] أدلة إدارة يونكس وجنو/لينكس تشرح بالتفصيل النواحي المتعلقة بالإدارة المحلية وإدارة أنظمة الطباعة.

[Gt] يمكننا هنا إيجاد معلومات محدّثة عن أنظمة الطباعة وإعداداتها، إضافة إلى تفاصيل عن بعض الطابعات. لتفاصيل محددة عن أصناف الطابعات ومشغلاتها، يمكننا الذهاب إلى http://www.linuxprinting.org.

[Koe] [Hin]يمكننا إيجاد معلومات عن أنظمة الملفات المختلفة المتاحة وأنماط إنشاء التقسيمات لتثبيت النظام.

مراجع أخرى للمعلومات بالعربيّة:

كتاب "إدارة العمليات والنظام" لصبري عبدالله.

إدارة الشبكة

د. رِمو سُـپي بُلدريتو

مقدمة

يستخدم نظام التشغيل يونكس (وجنو/لينكس) كمثال على معمارية الاتصال المعيارية. لقد أثبت يونكس منذ أسطورة UUCP (وهي Unix-to-Unix CoPy، وهي خدمة للنسخ بين أنظمة يونكس) وحتى الشبكات الحالية قابليته للاستخدام في العديد من المجالات المتعلقة بالاتصال وتبادل المعلومات. مع قدوم شبكات الحاسوب (الشبكات المحلية LAN، والشبكات واسعة النطاق WAN، وحتى أحدث الشبكات الإقليمية، والتي توفر اتصالاً بنقاط متعدّدة وبسرعات مختلفة (من 56Kb/s وحتى 1Gb/s [وأكثر])، نشأت الخدمات الحديثة المعتمدة على موافيق أسرع، والقابلة للنقل بين الحواسيب المختلفة، والمطوعة بشكل أفضل، مثل TCP/IP (برنامج التحكم بالنقل Transport Control Program / ميفاق الإنترنت Internet Protocol).

1 مقدمة في TCP/IP (حزمة TCP/IP)

يشكل ميفاق TCP/IP مثالاً على إرادة للاتصال وتوحيد التواصل على مستوى عالميّ.

إن TCP/IP – في الواقع – مجموعة من الموافيق الأساسيّة تمّت إضافتها إلى الميفاق الأساسيّ لتلبية الاحتياجات المختلفة في الاتصال بين أجهزة الحاسوب، مثل TCP, UDP, IP, ICMP, ARP.

TCP/IP هو الأكثر استخداماً بين معظم المستخدمين الحاليين للاتصال بحواسيب أخرى عن بعد (Telnet والصدفة الآمنة SSH)، واستخدام ملفات بعيدة (نظام ملفات الشبكة NFS)، أو لنقل الملفات (ميفاق نقل الملفات FTP، وميفاق النصوص التشعبية HTTP).

4.1 خدمات على TCP/IP

خدمات TCP/IP التقليدية الأكثر أهمية هي:

- **نقل الملفات:** يسمح ميفاق نقل الملفات FTP لمستخدم حاسوب بالحصول على ملفات أو إرسالها من حاسوب لآخر. لعمل ذلك، يجب أن يكون لدى المستخدم حساب في الحاسوب البعيد وأن يعرّف نفسه باسم حساب وكلمة مرور، أو يكون على المستخدم الاتصال بحواسيب تحوي مستودع معلومات (برمجيات، وثائق، إلخ) عبر حساب "المجهول" anonymous لقراءة ما في هذه الحواسيب عبر حاسوبه. هذا يختلف عن نظام ملفات الشبكة NFS الأحدث منه، (أو موافيق netbios على TCP/IP وهو "ابتكار" غير آمن مطلقاً في ويندوز، والذي يجب أن يستبدل ليأخذ ميفاق أقدم ولكن أكثر أمناً يدعى netbeui مكانه)، والذي يجعل من الممكن ضم نظام الملفات ظاهريًّا في الجهاز مما يمكن من الوصول إليه تفاعليًّا من حواسيب أخرى.

- **الاتصال (الولوج) عن بعد:** ميفاق الشبكة للطرفيات telnet يسمح للمستخدم بالوصول إلى حاسوب عن بعد. يستخدم الحاسوب المحلي كطرفية للحاسوب البعيد وكل شيء ينفّذ عليها، بينما يبقى الجهاز المحلي بعيداً عن متناول المستخدم الذي يبدأ الجلسة. تم استبدال هذه الخدمة الآن لتحل محلها خدمة الصدفة الآمنة SSH لأسباب أمنية يمكن

استخدام اتصال اتصال بعيد عبر telnet، وبهذا ترسل الرسائل على شكل نصّ صَرِف، وبعبارة أخرى، إذا كان أحد ما "يختبر" الرسائل على الشبكة، فهذا مثل النظر إلى شاشة المستخدم. يشفر SSH المعلومات (وهي ميزة إضافية إلى الاتصال)، وبهذا لا تتمكن أيّ نقطة خارجية على الشبكة من قراءة الحزم.

- **البريد الإلكتروني Email:** تجعل هذه الخدمة إرسال رسائل إلى مستخدمي حواسيب أخرى ممكناً. لقد صار هذا النوع من الاتصال عنصراً ضروريًّا للمستخدمين وتسمح بإرسال رسائل البريد إلى خادم مركزي، ومن ثم يمكن الحصول عليها مجدداً باستخدام برامج معينة (عملاء)، أو قراءتها عبر اتصال بالإنترنت.

إن التقدم التقني، والانخفاض المتزايد في أسعار الحواسيب يعني أن خدمات معينة قد تخصصت، وهي تضبط الآن على حواسيب معينة تعمل بنموذج الخادم – العميل. الخادم نظام يقوم بخدمات معينة لبقية الشبكة أو للعملاء المتصلين. والعميل حاسوب آخر يستفيد من هذه الخدمة. كل هذه الخدمات عموماً تقدّم ضمن TCP/IP:

- **أنظمة الملفات في أنظمة ملفات الشبكة:** تسمح لنظام بالوصول إلى الملفات عبر نظام بعيد بطريقة مضمّنة بشكل أفضل من FTP. تُصَدَّر أجهزة التخزين (أو أجزاء منها) إلى النظام الذي يرغب بـ"رؤية" الملفات كما لو كانت على أجهزة [تخزين] محلية. يسمح هذا الميفاق للخادم بإنشاء قواعد وطرق للوصول إلى الملفات، والتي (إذا تم ضبطها بشكل مناسب) تجعل الوصول إلى المكان الذي تقيم فيه البيانات فيزيائيًّا مستقلاً عن المكان الذي يتم فيه "الوصول" إلى المعلومات.

- **الطباعة عن بعد:** تسمح للمستخدمين بالوصول إلى الطابعات المتصلة بحواسيب أخرى.

- **التنفيذ عن بعد:** تسمح للمستخدم بتنفيذ برنامج على حاسوب آخر. هناك العديد من الطرق لتنفيذ برنامج بهذه الطريقة: إما عبر أمر (rsh, ssh, rexec)، أو عبر أنظمة فيها RPC (استدعاء إجراء عن بعد Remote Procedure Call)، والتي تسمح لبرنامج على حاسوب محلي بتنفيذ دالّة في برنامج على حاسوب آخر. لقد تمّت دراسة عمليات RPC بتعمّق وهناك عدد من التطبيقات، لكن أكثرها شيوعاً Xerox Courier و Sun RPC (هذا الأخير تمّ تبنيه في معظم أنظمة يونكس).

- **خوادم الأسماء:** في شبكات الحاسوب واسعة النطاق بيانات يجب أن تكون مركزية، ممّا يسهِّل استخدامها، مثل أسماء المستخدمين وكلمات المرور وعناوين الإنترنت وغيرها. كلّ هذا يسهّل على المستخدم بالحصول على حساب واحد لكلّ الأجهزة في المؤسّسة. على سبيل المثال، Yellow Pages من Sun (وهو NIS في إصدارات Sun الحالية) مصمّمة للتعامل مع كلّ هذه الأنواع من البيانات، وهي متاحة لمعظم أنظمة يونكس. نظام أسماء النطاقات DNS خدمة أخرى لأسماء النطاقات، ولكنها من النوع الذي يبقي علاقة مباشرة بين اسم المضيف واسم التعريف المنطقي لهذا الجهاز (عنوان IP).

- **خوادم الطرفية:** وصل طرفيات بخادم يشغّل telnet للاتصال بالحاسوب المركزي. هذه الأنواع من الإعدادات مفيدة بالإساس لتقليل التكاليف وتحسين الاتصالات بالحاسوب المركزي (في بعض الحالات).

- **خوادم الطرفيات الرسومية** (أنظمة النوافذ الموجهة للشبكة): وتسمح للحاسوب بتوضيح المعلومات رسومياً على جهاز عرض متصل بحاسوب آخر. الأكثر شيوعاً بين هذه الانظمة X Window.

4.2 ما هو TCP/IP ؟؟

إن TCP/IP في الحقيقة ميفاقا اتصال بين حواسيب مستقلة بعضها عن بعض.

فمن ناحية، يحدّد ميفاق التحكم بالنقل TCP قواعد الاتصال، وبهذا يتمكن حاسوب (مضيف) من التحدث لحاسوب آخر (إذا كنا نستخدم نموذج اتصالات OSI/ISO كمرجع، فهو يشكل الطبقة الرابعة، انظر إلى الجدول التالي).

TCP ميفاق موجَّه للاتصال، وبعبارة أخرى، يكافئ الهاتف، والتواصل يعتبر بأنه سيل البيانات.

ميفاق الإنترنت IP يشكل ميفاق تعريف الشبكات، ويُنشئ المسارات بين الحواسيب المختلفة.

وبعبارة أخرى، يوجِّه البيانات بين حاسوبين عبر الشبكات. وترادف الطبقة الثالثة من نموذج OSI/ISO، وهو ميفاق عديم الاتصال (انظر إلى الجدول التالي).

وكبديل عن TCP، لدينا UDP (اختصاراً لعبارة User Datagram Protocol)، والتي تتعامل مع البيانات كرسالة (datagrams) وترسل حزماً. وهو ميفاق عديم الاتصال (لا يشترط ان يكون الحاسوب المستقبل يستمع عندما يبدأ حاسوب آخر بالتواصل معه)، وله ميزة إنشاء حمل أقل على الشبكة من اتصال TCP، لكنه أقل موثوقية (فالحزم قد لا تصل، وقد تصل مكررة).

هناك ميفاق آخر بديل اسمه ICMP (اختصاراً لميفاق رسائل التحكم بالإنترنت Internet Control Message Protocol). يستخدم ICMP لرسائل التحكم والأخطاء. على سبيل المثال، اذا حاول شخص ما الاتصال بحاسوب مضيف، فقد يتلقى الحاسوب المحلي رسالة مفادها أنْ لا يمكن الوصول إلى المضيف. يمكن استخدام ICMP أيضاً لاستخلاص معلومات عن الشبكة. يشبه ICMP ميفاق UDP في أنه يتعامل مع رسائل (datagrams)، ولكنه أبسط من UDP، وذلك لأنها لا تحوي تعريفاً للمنفذ في ترويسة رسائلها (فالمنافذ التي تُرسل إليها هي المكان نفسه الذي تصدر منه وهو المكان الذي تقرأ فيه تطبيقات الخادم الحزم).

في نموذج اتصالات OSI/ISO (حيث OSI هو النموذج المرجعي للتواصل المتبادل بين الأنظمة المفتوحة open systems interconnection reference model، و ISO منظّمة المعايير الدولية International Standards Organization) نموذج نظري مطبق في العديد من الشبكات. هناك سبعة طبقات اتصال، ولكل منها واجهة للاتصال بما قبلها وما بعدها [من الطبقات].

الطبقة	الاسم	الاستخدام
7	طبقة التطبيقات	ميفاق نقل البريد البسيط SMTP، الخدمة نفسها.
6	طبقة التقديم	الاتصال عن بعد telnet، ويستخدم FTP ميفاق الخدمة
5	طبقة الجلسة	لا يكون مستخدماً عادةً
4	طبقة النقل	TCP, UDP النقل اعتماداً على ميفاق التواصل.
3	طبقة الشبكة	يجعل ميفاق الإنترنت IP توجيه الحزم ممكناً.
2	طبقة الربط	المشغّلات – التحويل إلى ما يوافق الميفاق الفيزيائيّ.
1	الطبقة الفيزيائيّة	Ethernet, ADSL, ... وترسل الحزم فيزيائيًّا.

وباختصار، TCP/IP مجموعة من المواثيق تشمل TCP, IP, UDP التي تقدم مجموعة من المزايا التحتية التي تستخدمها معظم التطبيقات.

بعض الموافيق التي تستخدم الخدمات المذكورة أعلاه صمّمتها Berkley و Sun ومؤسسات آخرى. تلك الموافيق غير مضمّنة (رسميًّا) كجزء من حزمة ميفاق الإنترنت (Internet Protocol Suit – IPS). ولكنها مبنية باستخدام TCP/IP، ولذلك تعتبر جزءاً رسميًّا من IPS. يمكن إيجاد وصف للميفاق متاح على الإنترنت في RFC 1011 (انظر إلى الملاحق عن [RFC [IET). هناك الآن إصدار جديد من الميفاق IPv6، والمسمى أيضاً Ipng (أي ميفاق الإنترنت – الجيل الجديد IP next generation) الذي يحل محل IPv4. يحسّن هذا الميفاق الموافيق السابقة في عناصر مثل وجود عدد أكبر من العُقَد [أو النقاط (كالحواسيب وغيرها) على الشبكة])، والتحكم بالنقل، والأمن، وتحسينات في التوجيه.

4.3 أجهزة الشبكة الفيزيائيّة (العتاد)

من وجهة النظر الفيزيائيّة (الطبقة الأولى في نموذج OSI)، فإن أكثر [أنواع] العتاد المستخدم شيوعاً للشبكات المحلية LAN تعرف باسم Ethernet (أو FastEthernet أو GigaEthernet). ومن مزاياها انخفاض تكلفتها، وسرعتها المقبولة (10 أو 100 أو 1000 ميجا بِت في الثانية)، وتثبيتها المريح للمستخدم.

> هناك ثلاثة أنواع من التوصيلات، وهي تعتمد على نوع الاتصال، فهناك الرفيعة والثخينة والأزواج المجدولة.

النوعان الأولان أثريان (وكانا يستخدمان الأسلاك المحوريّة)، بينما يستخدم الأخير أزواج مجدولة من الأسلاك والموصلات كتلك المستخدمة في الهواتف (وتعرف بالاسم RJ45). يعرف الاتصال عبر الأزواج المجدولة بالاسم 10baseT أو 100baseT (حسب السرعة) وتستخدم مُعيدات repeaters تعرف بالموزّعات المركزية hubs كنقطة توصيل. تستخدم تقنية Ethernet عناصر اتصال وسيطة (موزعات hubs، مبدّلات switches، موجّهات routers) لضبط عدّة أجزاء من الشبكة وتقسيم البيانات المارة فيها لتحسين أداء نقل البيانات. في المؤسسات الكبيرة عادة تكون شبكات Ethernet LAN هذه مرتبطة ببعضها عبر أسلاك الألياف الضوئيّة باستخدام تقنية واجهة البيانات الموزّعة بالألياف الضوئية FDDI (اختصاراً لعبارة Fiber Distributed Data Interface)، وهي أكثر كلفة وصعوبة في التركيب، ولكن بهذه التقنية يمكننا الحصول على سرعة نقل مكافئة لسرعة Ethernet دون محدودية المسافة المرتبطة بهذه الأخيرة (تسمح FDDI بمسافات تصل إلى كيلومترين اثنين). هذه التكلفة منطقية عند استخدامها بين بنائين أو أجزاء مختلفة من شبكة وكانت تلك الأجزاء مكتظة.

وفي نفس الوقت، هناك أنواع أخرى من العتاد أقل شيوعاً – ولكنها ليست أقل أهمية – مثل وضع النقل غير المتزامن ATM (اختصاراً لعبارة Asynchronous Transfer Mode) يسمح هذا العتاد بضبط شبكة LAN بمستوى عالٍ من جودة الخدمة وهو خيار جيد عندما نرغب بضبط شبكات ذات سرعات عالية وزمن وصول ضئيل، كتلك التي تتطلب نقلاً للفيديو بالوقت الحقيقي.

هناك أنواع عتاد أخرى يدعمها جنو/لينكس لتوصيل الحواسيب، يمكن أن نذكر منها: ترحيل الإطارات Frame Relay أو X.25 (تستخدم في الحواسيب التي تصل إلى أو توصل بين شبكات واسعة النطاق WAN أو للخوادم ذات الاحتياجات الكبيرة لنقل البيانات)، و Packet Radio (التوصيل عبر أمواج الراديو باستخدام موافيق مثل AX.25 أو NetRom أو Rose)، و خدمات الطلب الهاتفي التي تستخدم مودِمات الخطوط التسلسلية – وهي بطيئة، ولكنها ذات تكلفة منخفضة – التماثليّة أو الرقميّة (RDSL, DSL, ADSL، وغيرها). هذه الأخيرة هي الشائعة الاستخدام منزلياً أو في مجال الأعمال الصغيرة أو المتوسطة، وتتطلب ميفاقاً آخر لنقل الحزم، مثل SLIP أو PPP. لتبسيط مفهوم الأجهزة المختلفة في الشبكة، يستخدم TCP/IP عنواناً وهمياً للواجهة تتركز فيه كل الحزم التي سيرسلها جهاز فيزيائيّ واحد (التي تتضمّن شبكة أو جزءاً من شبكة). ونتيجة لذلك، فسيكون لدينا لكل جهاز اتصال في الحاسوب واجهة مرتبطة به في نواة نظام التشغيل.

مثال

في جنو/لينكس، تسمى شبكات Ethernet بالاسم ethX (حيث يحل محل X رقم يدل على ترتيب الجهاز، بدءاً بالرقم صفر)، وواجهات الخطوط التسلسلية pppX (لميفاق PPP) و slX (لميفاق SLIP)، أما fddiX فهو لشبكات FDDI. تستخدم هذه الأسماء في الأوامر لضبطها وإعطائها التعريف الذي سيسمح لها لاحقاً بالتواصل مع أجهزة أخرى على الشبكة.

في جنو/لينكس، قد يعني هذا أنه سيكون علينا تضمين الوحدات المناسبة للجهاز المناسب (بطاقة واجهة الشبكة) في النواة أو كوحدات [ملحقة]، وهذا يعني تصريف النواة بعد اختيار البطاقة المناسبة عبر make menuconfig مثلاً، بالإشارة إلى أنها مضمنة داخلياً أم كوحدة [ملحقة] (في الحالة الأخيرة يجب أيضاً تصريف [هذه] الوحدة).

يمكن رؤية جهاز الشبكة في مجلد /dev/، حيث يكون هناك ملف (ملف خاص يمكن أن يكون ملف كتل بيانات أو ملف محارف اعتماداً على نوع النقل) يمثل كل قطعة عتاد.

2 مفاهيم TCP/IP

كما رأينا، يشتمل التواصل على مجموعة من المفاهيم التي سنناقشها الآن:

- إنترنت/إنترانِت: يشير المصطلح "إنترانِت – intranet" إلى تطبيق تقنية الإنترنت (شبكة الشبكات) ضمن مؤسسة ، وبشكل أساسيّ لتوزيع المعلومات الداخلية للشركة وجعلها متاحة ضمن الشركة. فمثلاً، تشمل الخدمات التي يقدمها جنو/لينكس للإنترنت والإنترانت البريد الإلكتروني وخدمة التصفح (الموقع – www)، والأخبار، وغيرها.

- العُقدة (node): تشير العقدة (المضيف) إلى الجهاز المتصل بالشبكة (وبمفهوم أشمل، يمكن أن تكون العقدة جهاز حاسوب أو طابعة أو قارئ أقراص CD .. إلخ)؛ وبعبارة أخرى، هو عنصر فعال ويمكن تمييزه في الشبكة ويتطلب أو يقدم خدمة من نوع ما و/أو يتشارك بمعلومات.

- عنوان شبكة إثرنت (Ethernet Address أو MAC Address): هو رقم من 48 خانة ثنائية (مثل:

  بالنظام الست عشري – 00:88:40:73:AB:FF
  بالنظام الثنائيّ – 0000 0000 1000 1000 0100 0000 0111 0011 1010 1011 1111 1111

  الذي بداخل الجهاز الفيزيائيّ (العتاد) لمشغل شبكة إثرنت (بطاقة واجهة الشبكة)، ويسجّلها المصنّع (يجب أن يكون هذا العنوان الوحيد في العالم، ولكل مصنّع بطاقات شبكة نطاق محجوز مسبقاً).

- اسم المضيف: يجب أن يكون لكل عقدة في الشبكة اسم فريد أيضاً. يمكن ببساطة أن تكون أسماء، أو أن تكون أنماطاً مبنية على نمط تسمية نطاقات هرميًّا. يجب أن تكون أسماء العُقَد فريدة، وهذا سهل في الشبكات الصغيرة، وأكثر تعقيداً في الشبكات الأكبر، ومستحيل على الإنترنت مالم يتم تنفيذ نوع من التحكم. يجب أن تكون الأسماء ضمن حدٍّ أقصاه 32 رمزاً من بين الرموز a-z, A-Z, 0-9 [(أي الحروف الإنكليزية بالحجمين الصغير والكبير، والأرقام)]، ويجب أن تبدأ بحرف أبجدي، وأن لا تحوي فراغاً أو رمز '#'.

- عنوان إنترنت Internet Address – IP : ويتكون من أربعة أرقام ضمن النطاق 0 – 255 مفصولة بنقطة (مثل 192.168.0.1) وهو مستخدم علميًّا لتعريف الحواسيب على الشبكة أو على الإنترنت. تترجم الأسماء إلى عناوين IP

عبر خادم نظام أسماء النطاقات DNS، والذي يحول أسماء العُقد (التي يفهمها البشر) إلى عناوين IP (ينفِّذ الخدمة تطبيق اسمه named).

- المنفذ port : معرّف رقمي لصندوق البريد في عقدة يسمح لتطبيق معين بقراءة الرسائل ([التي قد تكون من نوع] TCP أو UDP) (على سبيل المثال، جهازان يتواصلان عبر telnet يفعلان هذا عبر المنفذ 23، ولكن إذا كان بينهما تناقل للبيانات عبر FTP، فسيفعلان هذا عبر المنفذ 21. قد يكون هناك تطبيقات مختلفة تتواصل بين عُقدتين عبر العديد من المنافذ المختلفة في نفس الوقت.

- عقدة الموجّه (العبّارة): هي نقطة تقوم بالتوجيه (نقل البيانات). يمكن للموجه – اعتماداً على مواصفاته – نقل المعلومات بين ميفاقي شبكة متماثلين أو مختلفين، ويمكن أيضاً أن يكون انتقائيًّا.

- نظام أسماء النطاقات DNS – Domain Name System : يجعل من الممكن التحقق من اسم واحد ويقدم إدارة لقواعد البيانات التي تقوم بالترجمة بين الاسم وعنوان إنترنت والمرتبة على شكل شجرة. ولعمل هذا، يتم تعريف النطاقات مفصولة بنقطة، حيث الأعلى (من اليمين لليسار) تصف تصنيفاً، شركة أو دولة (حيث يشير COM إلى نطاق تجاري، و EDU تعليمي، و GOV حكومي، و MIL عسكري (تابع للحكومة)، و ORG منظمة غير ربحية، وأي حرفين للإشارة إلى الدولة، أو حالات خاصة مثل CAT للإشارة إلى اللغة والثقافة الكَتلونية Catalan، وغيرها). المستوى الثاني يمثل المؤسسة، والقسم الثالث المتبقي يمثل الأقسام أو الدوائر أو التقسيمات بداخل تلك المؤسسة (مثل www.uoc.edu أو nteum@pirulo.remix.es). الاسمان الأولان (من اليمين لليسار)، وهما uoc.edu في الحالة الأولى و remix.es في الثانية يجب أن تُعطى (أو يوافَق عليها) من طرف SRI-NIC (وهي مؤسسة دولية تدير تسجيل نطاقات إنترنت، والبقية يمكن أن تعيّنها أو تضبطها المؤسسات [بنفسها].

- DHCP, bootp: هما ميفاقان يسمحان لعقدة عميل بالحصول على معلومات عن الشبكة (مثل عنوان IP للعقدة). العديد من المؤسسات ذات الأجهزة الكثيرة تستخدم هذه الآليّة لتسهيل إدارة الشبكات الكبيرة أو الشبكات التي فيها

مستخدمون متنقلون.

- ARP, RARP: في بعض الشبكات (مثل IEEE 802 LAN المتعارف عليها بالاسم إثرنت)، تكتشف عناوين IP آليًّا عبر استخدام عنصرين آخرين لحزمة ميفاق الإنترنت: ميفاق حلّ العناوين ARP، وميفاق حلّ العناوين العكسيّ RARP. يستخدم ARP الرسائل العامة لتحديد عناوين إثرنت (معيار MAC للطبقة الثالثة من من نموذج OSI) المرتبط بعنوان معين من طبقة الشبكة (عنوان IP). يستخدم RARP الرسائل العامة (الرسائل التي تصل إلى كل العقد) لتحديد العنوان من طبقة الشبكة المرتبط بعنوان عتاد معين. RARP ضروريّ بشكل خاص للأجهزة عديمة الأقراص، والتي بالعادة لا تكون فيها عناوين طبقة الشبكة معروفة عند الإقلاع.

- مكتبة المقابس Socket Library: في يونكس، TCP/IP كلّه منفّذ كجزء من نواة نظام التشغيل (سواء كان مدمجاً بها أو كوحدة يتم تحميلها عندالإقلاع، كما هي حالة مشغّلات الأجهزة في جنو/لينكس).

الطريقة التي يستخدمها بها مبرمج هي عبر واجهة برمجة التطبيقات API التي تتضمّن واجهة المصدر البرمجي هذه. لميفاق TCP/IP، الواجهة الأكثر شيوعاً هي Berkeley Socket Library (يستخدم وندوز مكتبة مكافئة اسمها Winsocks). تجعل هذه المكتبة إنشاء نقطة اتصال نهائية (مقبس socket)، وربطها بوحدة بعيدة ومنفذ ما (bind)، وتقديم خدمة الاتصال (عبر connect, listen, accept, send, sendto, recv, recvfrom, مثلاً) أمراً ممكناً. تتيح هذه المكتبة أيضاً وضع اتصال أكثر عموميّة (من عائلة AT INET)واتصالات محسنة أكثر لحالات تتصل فيها العمليات داخل نفس الجهاز (عائلة AF UNIX). في جنو/لينكس، مكتبة المقابس جزء من مكتبة C القياسية libc، وتدعم AF_INET، و AF_UNIX، و AF_IPX (لموافيق Novell)، و AF_X25 (لموافيق X.25)، و AF_ATMPVC و AF_ATMSVC (لميفاق ATM)، و AF_AX25، و AF_NETROM، و AF_ROSE (لميفاق amateur radio).

3 كيفية إعطاء عنوان IP

يعطى العنوان لبطاقة الشبكة، ولهذا العنوان جزءان. الجزء الذي على اليسار يمثل تعريف الشبكة، والجزء على اليمين يمثل تعريف العُقدة. بأخذ النقطة المذكورة أعلاه بعين الاعتبار (أربعة أرقام بين 0 و 255، أو 32 بت، أو 4 بايت)، يمثل كل بايت إما الشبكة أو العُقدة. تُعيّن البطاقة الشبكة، وتعيّن المؤسسة (أو المزوّد) العُقدة.

هناك بعض القيود: 0 (على سبيل المثال: 0.0.0.0) في مساحة الشبكة محجوز للتوجيه مبدئيًا، و 127 (كما في: 127.0.0.1) محجوز للجهاز المحلي (local loopback أو local host)، و 0 في الجزء المتعلق بالعُقدة يشير إلى الشبكة (192.168.0.0)، و 255 محجوز لإرسال الحُزم إلى عنوان broadcast، أي إلى كل الأجهزة (192.168.255.255). قد يكون هناك أنواع مختلفة من الشبكات أو العناوين في أنظمة العناوين المختلفة:

المجموعة A (ويكون فيها الجزء الأول من اليسار للشبكة والثلاثة الباقية للمضيف، أي network.host.host.host): وهي من 1.0.0.1 وحتى 126.254.254.254 (أي 126 شبكة تتسع كل منها لأكثر من 16 مليون عُقدة)، وتشير للشبكات الكبيرة. المعيار الثنائي هو: 0 + 7 بت للشبكة + 24 بت للعُقدة.

المجموعة B (ويكون فيها جزءان للشبكة وآخران للمضيف، أي net.net.host,host): وتأخذ العناوين من 128.1.0.1 وحتى 191.255.254.254 (16 ألف شبكة لكل منها 65 ألف عُقدة)؛ عادة ما يكون الجزء الأول من عنوان العُقدة مستخدماً لتعريف الشبكات الفرعية بداخل المؤسسة. المعيار الثنائي هو 10 + 14 بت للشبكة + 16 بت للعُقدة.

المجموعة C (وتكون net.net.net.host): وتأخذ العناوين 192.1.1.1 وحتى 223.255.255.254 (مليوني شبكة ذات 254 عقدة). المعيار الثنائي هو 110 + 21 بت للشبكة + 8 بت للعقدة.

المجموعتان D و E (وتكونان net.net.net.host): وتأخذان العناوين 224.1.1.1 وحتى 255.255.255.254، وهي

محجوزة للإرسال من عقدة إلى مجموعة عقد تشكل جزءاً من المجموعة (multicast) وللتجارب.

بعض نطاقات العناوين تم حجزها كي لا تكون جزءاً من شبكات عامة، وتعتبر شبكات خاصة (حواسيب متصلة دون اتصال خارجي؛ الرسائل لن يتم إرسالها عبر الإنترنت، وإنما ضمن إنترانت). نطاقات هذه العناوين في المجموعة A هي 10.0.0.0 – 10.255.255.255، وفي المجموعة B هي 172.16.0.0 – 172.31.0.0، وفي المجموعة C تكون 192.168.0.0 – 192.168.255.0.

للبثّ broadcast عنوان خاص، وذلك لأن كل عقدة في الشبكة تستمع إلى كلّ الرسائل (إضافة إلى عنوانها الخاص). يجعل هذا العنوان إرسال رسائل datagrams (وعادة تكون معلومات توجيه أو رسائل تحذير) إلى الشبكة، وستتمكن كل العقد في الشبكة من قراءتها. على سبيل المثال، عندما يحاول ARP إيجاد عنوان إثرنت المتعلق بعنوان IP يقوم ببث رسائلة على هذا العنوان، فيتم إرسالها إلى كلّ الأجهزة على الشبكة في نفس الوقت. تقرأ كل عقدة في الشبكة هذه الرسالة وتقارن عنوان IP الذي يتم البحث عنه، ثم ترسل ردًّا إلى العقدة المرسلة إذا تطابق العنوان.

هناك مفهومان مرتبطان بالنقطة المذكورة أعلاه، وهما الشبكات الفرعية والتوجيه بين هذه الشبكات الفرعية. تقسّم الشبكات الفرعيةُ الجزءَ المتعلق بالعقد إلى شبكات أصغر ضمن الشبكة نفسها، وذلك لتحسين مرور البيانات مثلاً. الشبكة الفرعية مسؤولة عن إرسال البيانات إلى نطاقات عناوين IP معينة، وما هذا إلا امتداد لمفاهيم الشبكات A و B و C نفسها، ولكن بتطبيق عملية إعادة التوجيه هذه في الجزء المتعلق بعنوان العقدة. يقدم قناع الشبكة عدد الخانات الثنائية المترجمة كمعرفات شبكة فرعية، وقناع الشبكة هذا رقم من 32 خانة ثنائية (تماماً كعنوان IP). للحصول على معرف الشبكة الفرعية، علينا أن نقوم بعملية "و" (AND) المنطقية بين القناع والعنوان، والتي ستقدم لنا عنوان الشبكة الفرعية. فعلى سبيل المثال، مؤسسة لديها شبكة من النوع B تحمل الرقم 172.17.0.0، فسيكون لديها قناع شبكة يحمل الرقم 255.255.0.0. تتكون هذه الشبكة داخلياً من شبكات أصغر (واحدة في كل طابق من المبنى على سبيل المثال). بهذه الطريقة، نطاق العناوين مقسم إلى 20 شبكة فرعية (بعدد الطوابق، عدا 172.17.1.0 والتي لها دور خاص هنا)، وهي 172.17.1.0 إلى 172.17.20.0. النقطة التي تصل كل تلك الطوابق – ويطلق عليها السَنَد (بالإنجليزية: backbone) – لها عنوانها الخاص، ويكون مثلاً 172.17.1.0.

تتشارك هذه الشبكات الفرعية بنفس عنوان IP للشبكة، بينما يستخدم الجزء الثالث لتحديد كلّ من الشبكات الفرعية بداخلها (ولهذا ستستخدم قناع الشبكة 255.255.255.0).

المفهوم الثاني – وهو التوجيه – يمثل الطريقة التي ترسل بها الرسائل عبر الشبكات الفرعية. على سبيل المثال، لنفترض أن هناك ثلاثة دوائر [في مؤسسة ما] لديها شبكات إثرنت فرعية:

1) المشتريات (الشبكة الفرعية 172.17.2.0).

2) العملاء (الشبكة الفرعية 172.17.4.0).

3) الموارد البشرية (الشبكة الفرعية 172.17.6.0).

4) السَنَد وهو متصل عبر FFDI (الشبكة الفرعية 172.17.1.0).

من أجل توجيه الرسائل بين الحواسيب على الشبكات الثلاثة، فسنحتاج إلى ثلاثة عبّارات لكلّ منها بطاقتا شبكة للتبديل بين إثرنت و FFDI. وستكون كالآتي:

1) عناوين عبّارة دائرة المبيعات: 172.17.2.1 و 172.17.1.1.

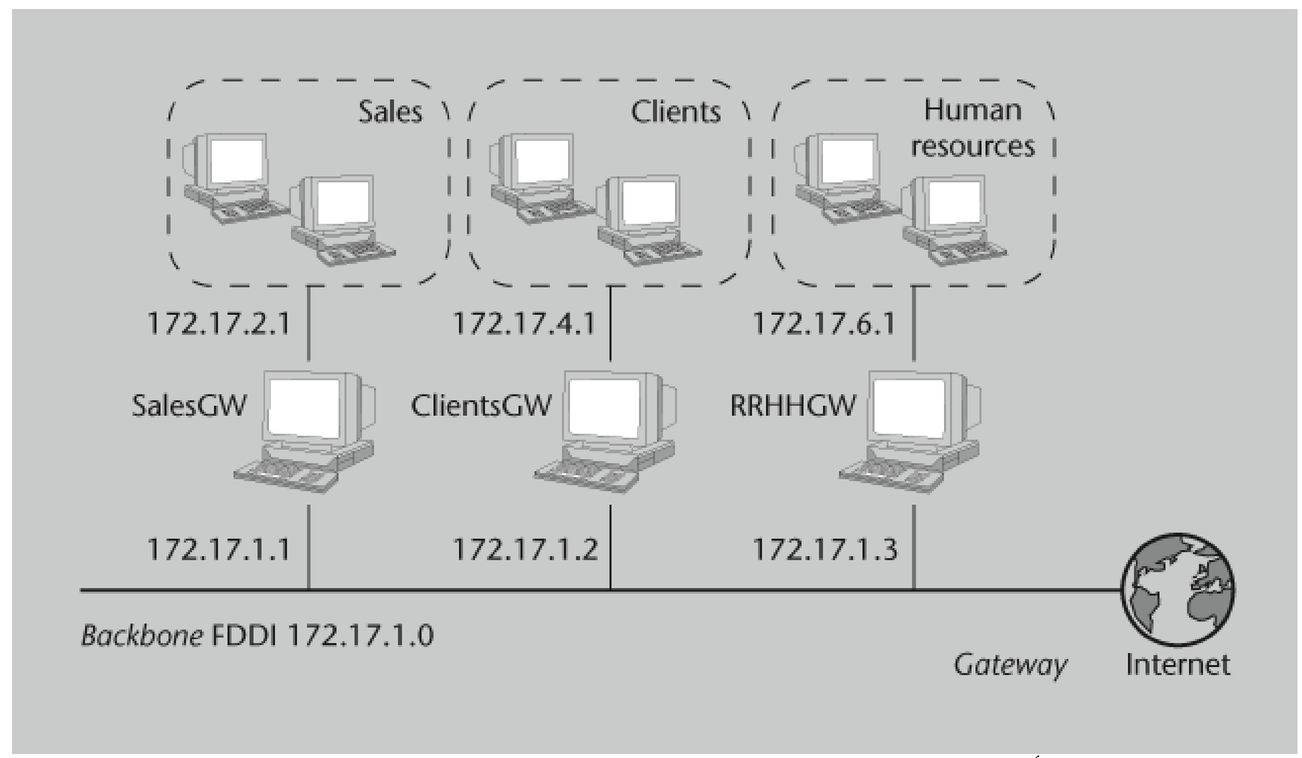
2) عناوين عبّارة دائرة شؤون العملاء: 172.17.4.1 و 172.17.1.2.

3) عناوين عبارة الموارد البشرية: 172.17.6.1 و 172.17.1.3

وبعبارة أخرى، عنوان من جهة الشبكة الفرعية، وآخر من جهة السَنَد.

عند إرسال الرسائل بين الأجهزة في منطقة المبيعات، فليس من الضروري أن تغادر العبّارة، حيث سيجد TCP/IP الجهاز مباشرة. تظهر المشكلة عندما يرغب جهاز ما من قسم المشتريات (على فرض أن اسمه Purchases0) بإرسال رسالة إلى

جهاز آخر من قسم الموارد البشرية (ليكن HumanResources3). يجب أن تمر الرسالة من خلال العبّارتين ذاتي العلاقة. عندما يرى Purchases0 أن HumanResources3 موجود في شبكة أخرى، فسيقوم بإرسال الرسالة إلى عبّارة المشتريات، والتي بدورها ترسلها إلى عبّارة الموارد البشرية، والتي بدورها ترسلها إلى الجهاز HumanResources3. الفائدة من وجود الشبكات الفرعية واضح، آخذين بعين الاعتبار أن مرور البيانات بين أجهزة قسم المشتريات على سبيل المثال لن يؤثر على أجهزة قسمي الموارد البشرية والعملاء (لكنها أعقد وأكثر كلفة فيما يتعلق بتصميم وبناء الشبكة).



شكل 1: ضبط الأجزاء والعبارات في الإنترانت

يستخدم ميفاق الإنترنت جدولاً لتوجيه الحزم بين الشبكات المختلفة، والذي به عنوان توجيه مبدئيّ مرتبط بالشبكة 0.0.0.0. تقترن كل هذه العناوين بعنوان واحد، حيث لا يهم أي من هذه الـ 32 بت؛ ويتم إرسالها عبر العبّارة الافتراضية إلى الشبكة المقصودة. في عبّارة المشتريات مثلاً، يمكن أن يكون الجدول كالتالي:

العنوان	قناع الشبكة	العبّارة	الواجهة
172.17.1.0	255.255.255.0	–	fddi0
172.17.4.0	255.255.255.0	172.17.1.2	fddi0
172.17.6.0	255.255.255.0	172.17.1.3	fddi0
0.0.0.0	0.0.0.0	172.17.2.1	fddi0
172.17.2.0	255.255.255.0	–	eth0

تعني الإشارة "-" أن الجهاز متصل مباشرة ولا يحتاج توجيهاً. يتم الإجراء المتبع لتحديد ما إذا كان التوجيه مطلوباً أم لا بتنفيذ عملية بسيطة جدًّا مكونة من عمليتي "و" منطقيتين (بين الشبكة الفرعية و القناع، وبين المصدر والقناع) ومقارنة النتيجتين. إذا تطابقتا فليس هناك توجيه، ولكن الجهاز المعرف كعبارة يجب أن يرسَل لكل جهاز لكي يوجّه هذا الجهاز الرسالة.

فمثلاً، رسالة مرسلة من 172.17.2.4 إلى 172.17.2.6 تعني:

172.17.2.4 AND 255.255.255.0 = 172.17.2.0
172.17.2.6 AND 255.255.255.0 = 172.17.2.0

وبما أن النتيجتين متشابهتان، فلن يكون هناك توجيه. ومن ناحية أخرى، إذا فعلنا نفس الشيء بين 172.17.2.4 و 172.17.6.6 فسنرى أنه سيكون هناك توجيه عبر 172.17.2.1 مع تغيير في الواجهة (من eth0 إلى ffdi0) إلى 172.17.1.1، ومن هناك إلى 172.17.1.2 مع تغيير آخر في الواجهة (ffdi0 إلى eth0)، ومن ثم إلى 172.17.6.6. سيستخدم التوجيه المبدئي عندما لا يكون هناك أيّة قوانين مطابقة. إذا تطابق قانونان، فالقانون الذي يطابق بشكل أدق – وبعبارة أخرى، القانون ذي العدد الأقل من الأصفار – سيُستخدَم. من أجل بناء جداول التوجيه، يمكننا استخدام الأمر route أثناء تشغيل الجهاز؛ لكن إذا كان من الضروري استخدام قواعد أكثر تعقيداً (أو توجيهاً آليًّا)، فيمكننا استخدام أمر ميفاق معلومات التوجيه Routing Information Protocol – RIP، أو ميفاق العبّارة الخارجية External Gateway Protocol – EGP، أو ميفاق العبارة الحدودية Border Gateway Protocol – BGP. هذه الموافيق موجودة ضمن الأمر gated.

> لتركيب جهاز في شبكة موجودة، من الضروري أن تكون لدينا المعلومات التالية – بالحصول عليها من مقدم الخدمة أو من مديرها: عنوان IP للعقدة، عنوان IP الشبكة، عنوان البثّ، عنوان قناع الشبكة، عنوان الموجّه، عنوان DNS.

إذا كنا نضبط شبكة لن يكون لها اتصال بالإنترنت مطلقاً، فيمكننا استخدام العناوين التي نريدها، ولكن يُنصح بالإبقاء على ترتيب مناسب موافق لحجم الشبكة التي ستُستخدَم، وذلك لتجنب المشاكل الإدارية بداخل الشبكة المعنيّة. سنرى الآن كيف نعرّف الشبكة والعقد لشبكة خاصة (علينا أن نكون حذرين، وذلك لأنه إذا كان الجهاز متصلاً بالشبكة، فقد نزعج مستخدماً آخر تم إعطاؤه

هذا العنوان): عنوان العقدة 192.168.110.23، قناع الشبكة 255.255.255.0، الجزء المتعلق بالشبكة 192.168.110، الجزء المتعلق بالعقدة 23، عنوان الشبكة 192.168.110.0، عنوان البث 192.168.110.255.

4 كيفية ضبط الشبكة

4.1 ضبط متحكم واجهة الشبكة

بمجرد تحميل نواة جنو/لينكس، فإنها تنفّذ الأمر init، والذي بدوره يقرأ ملف الإعدادات etc/inittab/ ويبدأ بعملية الإقلاع. وبشكل عام، يكون في ملف inittab سلاسل مثل: si::sysinit:/etc/init.d/boot، وهي سلاسل الإقلاع. وعامّةً يوم هذا النصّ البرمجي باستدعاء النصوص البرمجية الأخرى والتي تشمل نصّ تشغيل الشبكة.

مثال

في دبيان، يتم تنفيذ الأمر etc/init/network/ لضبط واجهة الشبكة، اعتماداً على مستوى الإقلاع؛ على سبيل المثال، في مستوى الإقلاع الثاني، سيتم تنفيذ كلّ الملفات التي تبدأ بحرف S في المجلد /etc/rc2.d/ (والتي تكون روابط لمجلد /etc/initd/)، وفي عملية الإيقاف، يتم تنفيذ كل الملفات التي تبدأ بلحرف K في نفس المجلد. بهذه الطريقة، النص البرمجي موجود مرة واحدة (في /etc/init.d/)، ويتم إنشاء رابط في المجلد المتعلق بحالة العقدة اعتماداً على الخدمة المطلوبة في تلك الحالة.

يتم إنشاء أجهزة الشبكة تلقائيًّا عند تشغيل العتاد المتعلق بها. فمثلاً، يُنشئ مشغّل الإثرنت الواجهات eth[0...n] على الترتيب عند توصيل العتاد المتعلق بها.

يمكن أن تضبط واجهة الشبكة في تلك اللحظة، وهذا يتطلب خطوتين: إعطاء عنوان الشبكة للجهاز، وإقلاع معاملات الشبكة في النظام. الأمر المستخدم لهذا هو ifconfig (اختصاراً لعبارة interface configure). ومن الأمثلة عليه:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
```

مما يشير إلى أنه يفترض أن يكون الجهاز eth0 مضبوطاً بعنوان IP هو 92.168.110.23 وقناع الشبكة 255.255.255.0. تشير كلمة up إلى أنه سيتم تفعيل الواجهة (لتعطيلها نفّذ ifconfig eth0 down)، إذا لم يتم تحديد قيم، يفترض الأمر أن عليه استخدام القيم المبدئية. في المثال السابق، ستضبط النواة هذا الجهاز على أنه ضمن شبكة من المجموعة C بالعنوان 192.168.110.23 وعنوان البث 192.168.110.255.

هناك أوامر مثل ifup و ifdown تجعل من الممكن ضبط وإلغاء ضبط الشبكة ببساطة أكثر باستخدام الملف /etc/network/interfaces للحصول على المعاملات الضرورية (راجع man interfaces لمعرفة الصيغة).

في دبيان، هناك طريقة أخرى أبسط لضبط الشبكة (تعتبر ذات مستوى عالٍ)، وتستخدم الأمرين المذكورين أعلاه ifup و ifdown والملف /etc/network/interfaces. إذا قررنا استخدام هذه الأوامر، فعلينا أن لا نضبط الشبكة على مستوى منخفض، حيث هذه الأوامر كافية لضبط وإلغاء ضبط الشبكة.

من أجل تعديل معاملات شبكة الواجهة eth0، فيمكننا [استخدام الأوامر التالية] (لمزيد من المعلومات راجع man interfaces في القسم 5 من أدلة استخدام يونكس المضمنة في نظام التشغيل):

لكل خدمات الشبكة على eth0	ifdown eth0
تحرير وتعديل معاملات الشبكات والواجهات	vi /etc/network/interfaces
بدء خدمات الشبكة على eth0	ifup eth0

لنفترض أننا نرغب بضبط الواجهة eth0 في دبيان، والتي لها عنوان IP ثابت 192.168.0.123، ولدينا العبارة 192.168.0.1. علينا تحرير الملف /etc/network/interfaces، وذلك لكي تتضمن قسماً مثل:

```
iface eth0 inet static
        address 192.168.0.123
        netmask 255.255.255.0
        gateway 192.168.0.1
```

إذا كنا ثبتنا حزمة resolvconf، فيمكننا إضافة سطور لتحديد معلومات DNS. على سبيل المثال:

```
iface eth0 inet static
        address 192.168.0.123
        netmask 255.255.255.0

        gateway 192.168.0.1
        dns-search remix.org
        dns-nameservers 195.238.2.21 195.238.2.22
```

بعد تفعيل الواجهة، فإن معاملات سطر الأوامر للخيارات dns-search و dns-nameservers تصير متاحة ليتم تضمينها في resolv.conf. معامل سطر الأوامر remix.org للخيار dns-search مرتبط بمعامل خيار البحث في resolv.conf

(سنبحث في هذا بتعمق أكثر لاحقاً)، والمعاملات 195.238.2.21 و 195.238.2.22 للخيار dns-nameserver ترتبط بمعاملات الخيار nameserver في resolv.conf (راجع man resolv.conf). يمكن أيضاً ضبط الشبكة على مستوى منخفض عبر الأمر ip (المكافئ للأمرين ifconfig و route)، لكن هذا الأمر أقوى وذو استخدامات أكثر بكثير (يمكن استخدامه لإنشاء tunnels، ولتغيير التوجيه، إلخ)، وهو أكثر تعقيداً وينصح باستخدام الإجراءات السابقة للضبط الأساسيّ للشبكات.

4.1.1 ضبط الشبكة بأسلوب فيدورا

تستخدم ردهات وفيدورا بنية ملفات مختلفة لضبط الشبكة: etc/sysconfig/network/. على سبيل المثال، لضبط الشبكة ضبطاً ثابتاً:

NETWORKING=yes HOSTNAME=my-hostname FORWARD_IPV4=true GATEWAY="XXX.XXX.XXX.YYY"	 اسم المضيف المضبوط بالأمر hostname true لعبارات جدران الحماية لشبكات NAT false لأيّ حالة أخرى العبارة التي تقود إلى الخارج (إلى الإنترنت)

للضبط باستخدام DHCP، من الضروري حذف السطر GATEWAY، حيث سيقوم الخادم بتعيينه. وإذا كان سيتم استخدام NIS، فيجب إضافة سطر باسم النطاق: NISDOMAIN=NISProject1

لضبط الواجهة eth0 في الملف etc/sysconfig/network-scripts/ifcfg-eth0/:

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=XXX.XXX.XXX.255
IPADDR=XXX.XXX.XXX.XXX
NETMASK=255.255.255.0
NETWORK=XXX.XXX.XXX.0
ONBOOT= yes        يفعل الشبكة عند الإقلاع
```

من الإصدار الثالث لفيدورا وصاعداً، صار من الممكن أيضاً إضافة:

```
TYPE=Ethernet
HWADDR=XX:XX:XX:XX:XX:XX
GATEWAY=XXX.XXX.XXX.XXX
IPV6INIT=no
USERCTL=no
PEERDNS=yes
```

أما في حال الضبط لاستخدام DHCP:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

لتعطيل DHCP، غير BOOTPROTO=dhcp إلى BOOTPROTO=none. لأيّ تغيير في هذه الملفات يجب أن نعيد تشغيل الخدمات باستخدام service network restart (أو /etc/init.d/network restart).

يجب اتخاذ الخطوات الثلاثة التالية لتغيير اسم المضيف:

1) الأمر hostname new-name

2) غيّر إعدادات الشبكة في /etc/sysconfig/network بتحرير HOSTNAME=new-name.

3) استعادة كل الخدمات (أو إعادة التشغيل):

- service network restart (أو تنفيذ /etc/init.d/network restart).

- إعادة تشغيل سطح المكتب بالتحويل إلى الطور النصي init 3 ثم الانتقال إلى الطور الرسومي init 5.

بالتحقق إذا ما كان الاسم غير مسجل في /etc/hosts. يمكن تغيير اسم المضيف أثناء وقت التنفيذ عبر

```
sysctl -w kernel.hostname=”newname”
```

4.1.2 ضبط الشبكة اللاسلكية

لضبط واجهة الشبكة اللاسلكية، نستخدم بالأساس الحزمة wireless-tools[1] (إضافة إلى ifconfig و ip). تستخدم هذه الحزمة الأمر iwconfig لضبط واجهة الشبكة اللاسلكية، لكن يمكن عمل هذا أيضاً عبر /etc/network/interfaces.

مثال: ضبط الشبكة اللاسلكية في دبيان (تشبه طريقة فيدورا):

لنفترض أننا نرغب بضبط بطاقة شبكة لاسلكية Intel Pro/Wireless 2200BG (شائعة جداً في العديد من الحواسيب المحمولة، مثل Hp، Dell، …). البرمجية التي تتحكم بالبطاقة تقسم عادة إلى جزئين: الوحدة البرمجية التي يتم تحميلها في النواة عبر الأمر modprobe، والإطار البرمجي وهو الكود الذي يتم تحميله في البطاقة والذي يعطيه لنا المصنّع (راجع موقع إنتل لهذا الصنف). بما أننا نناقش الوحدات، فمن المفيد استخدام الحزمة module-assistant في دبيان والتي تسمح لنا بإنشاء وتثبيت الوحدات بسهولة (ولدينا خيار آخر وهو تثبيت المصادر البرمجية وإنشاء الوحدات ذات العلاقة). سنصرّف ونثبت البرمجية (التي يمكننا إيجادها على موقع المصنّع واسمها ipw2200) باستخدام الأمر m-a في حزمة module-assistant.

```
apt-get install module-assistant (لتثبيت الحزمة)
m-a -t update
m-a -t -f get ipw2200
m-a -t --build ipw2200
m-a -t install ipw2200
```

يمكننا تنزيل إصدار الإطار البرمجي المتوافق من عنوان الموقع الذي يوفره المصنّع (في توثيق المنتَج) إضافة إلى إصدار المشغّل الذي نحتاجه، وهو في حالتنا هذه إصدار المشغّل 1.8 وإصدار الإطار البرمجي 2.0.4، ويتم الحصول عليها من العنوان التالي:

http://ipw2200.sourceforge.net/firmware.php

علينا بعد ذلك أن نفك الضغط ونثبت الإطار البرمجي:

```
tar xzvf ipw2200fw2.4.tgz C /tmp/fwr/
cp /tmp/fwr/*.fw /usr/lib/hotplug/firmware/
```

هذا سينسخ ثلاثة حزم (وهي ipw2200-bss.fw و ipw2200-ibss.fw و ipw2200-sniffer.fw)، ثم يتم تحميل الوحدة كالتالي: modprobe ipw2200، ثم إعادة تشغيل الجهاز، وبعدها – من سطر الأوامر – يمكننا تنفيذ الأمر dmesg | grep ipw الذي سيظهر لنا بعض السطور المشابهة لما هو أدناه والتي تشير إلى أنه تم تحميل النواة (يمكن فحص ذلك بالأمر lsmod):

```
ipw2200: Intel(R) PRO/Wireless 2200/2915 Network Driver, git1.0.8
ipw2200: Detected Intel PRO/Wireless 2200BG Network Connection
...
```

علينا بعد ذلك تنزيل حزمة أدوات الشبكة اللاسلكي التي تحوي iwconfig وذلك لتثبيت أدوات الشبكة اللاسلكية عبر

---

1 هذه الحزمة صارت أثرية الآن، وقد تم نقل مزاياها بالكامل إلى أدوات أخرى، ومنها الأداة الأحدث التي كتبت لتحل محلها ولتغطي كل احتياجات مستخدم الشبكات اللاسلكية في أنظمة جنو/لينكس وهي أداة iw (راجع man iw).

apt-get – إضافة إلى غيرها –، وإذا نفّذنا iwconfig، سيظهر شيء يشبه ما يلي:

```
eth1     IEEE    802.11b          ESSID:"Name-of-the-Wifi"
Mode: Managed          Frequency:2.437 GHz
Access Point: 00:0E:38:84:C8:72
Bit Rate=11 Mb/s        TxPower=20 dBm
Security mode: open
...
```

يجب علينا عند ذلك ضبط ملف الشبكة، [وذلك عبر gedit /etc/network/interfaces – على سبيل المثال –، وإضافة واجهة الشبكة اللاسلكية eth1[2]، على سبيل المثال:

```
iface eth1 inet dhcp
        pre-up iwconfig eth1 essid "اسم الشبكة اللاسلكية"
        pre-up iwconfig eth1 key open XXXXXXXXXX
```

تنفِّذ سطور pre-up الأمر iwconfig قبل تفعيل الواجهة. يستخدم هذا الإعداد إذا رغبنا باستخدام الخدمة في وضع DHCP (الحصول على عناوين IP آليًّا، كما سنرى). بدلاً من DHCP، علينا استخدام الكلمة static و يجب إدخال السطور التالية – على سبيل المثال – (كما في البطاقة السلكية):

```
address 192.168.1.132
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.1.1
```

وهناك طريقة أخرى لضبط الواجهة وهي:

```
iface eth1 inet dhcp
    wireless-essid "اسم الشبكة اللاسلكية"
    wireless-key 123456789e
```

يمكننا بعد ذلك تشغيل الشبكة بالأمر ifup eth1 وسنعطى معلومات عن الاتصال وحالة وجودة الاستقبال. من أجل

---

2 في معظم الحالات ستكون بطاقة الشبكة اللاسلكية معرفة، ولن تحتاج للقيام بمعظم ما سبق بنفسك. تسمية بطاقة الشبكة اللاسلكية يمكن أن تتبع نمط تسمية الشبكات السلكية eth0, eth1, ...ethN، ويمكن أن تتبع نمطاً آخر مثل wifi0, wifi1, ... أو wlan0, wlan1, ... لذا تأكد من الحالة لديك قبل المتابعة.

تفحّص الشبكات اللاسلكية المتاحة (نقاط الوصول)، يمكننا استخدام iwlist scan، والتي ستعطينا معلومات عن الشبكات المتاحة، وإذا رغبنا بالاتصال بشبكة أخرى، فيمكننا استخدام الأمر iwconfig لتغيير الشبكة أو نقطة الوصول.

4.2 ضبط محوّل الأسماء

الخطوة التالية ضبط محول الأسماء، والذي يغير الأسماء مثل pirulo.remix.com إلى 192.168.110.23. الملف /etc/resolv.conf مستخدم لهذا الغرض. الصيغة بسيطة جداً (سطر نصي واحد لكل جملة). هناك ثلاث كلمات مفتاحية لهذا الغرض: domain (النطاق المحلي)، search (عرض قائمة بالنطاقات البديلة)، و name server (عنوان IP لخادم أسماء النطاقات DNS).

مثال على /etc/resolv.conf

```
domain remix.com
search remix.com piru.com
name server 192.168.110.1
name server 192.168.110.65
```

تعتمد قائمة خوادم الأسماء هذه على بيئة الشبكة التي قد تتغير اعتماداً على المكان الذي يتواجد فيه الجهاز والمكان الذي يتصل به. البرنامج المستخدم للاتصال بخطوط الهاتف (pppd) أو الحصول على عنوان IP آليًّا (dhclient) يمكن أن يعدّل resolv.conf لإضافة أو حذف خوادم؛ لكن هذه المميزات لا تعمل دائماً بشكل صحيح، ويمكن أحياناً أن تنشئ تعارضات أو إعدادات غير صحيحة. تصلح حزمة resolvconf المشكلة كما ينبغي وتسمح لنا بضبط خوادم الأسماء بسهولة وبشكل متغير. resolvconf مصممة للعمل دون أن يضطر المستخدم لأن يضبط أي شيء يدويًّا، رغم هذا، فالحزمة جديدة نوعاً ما وقد تحتاج مساعدة من العمل اليدوي لجعلها تعمل كما يجب. لمزيد من المعلومات:

http://packages.debian.org/unstable/net/resolvconf

وهناك ملف آخر هام هو etc/host.conf/، والذي يمكن أن يستخدم لضبط سلوك محول الأسماء. هذا الملف هام جداً لأنه يشير إلى المكان الذي يحوّل فيه اسم أو عنوان العقدة أولاً. يمكن مراجعة هذا في خادم DNS أو الجداول المحلية بداخل الجهاز الحاليّ (etc/hosts/).

مثال على etc/hosts/

```
order hosts,bind
multi on
```

يشير هذا الإعداد إلى أنه يفترض أن يتم التحقق بالرجوع أولاً إلى خادم DNS، ويشير أيضاً (في السطر الثاني) إلى أنه يفترض إرجاع كلّ العناوين الصالحة الموجودة في etc/hosts/. ونتيجة لذلك، فالملف etc/hosts/ هو المكان الذي توضع فيه العناوين المحلية ويمكن أن تستخدم أيضاً للوصول إلى العقد دون الحاجة للرجوع إلى خادم DNS.

عملية المراجعة أسرع بكثير، ولكن العيب فيها هو أنه إذا تغيرت العقدة، فسيكون العنوان غير صحيح. في نظام مضبوط بشكل صحيح، يجب أن تظهر فقط العقدة المحلية ومدخلة لواجهة الإرجاع loopback.

مثال على etc/hosts/

127،0،0،1	localhost	loopback
192،168،1،2	pirulo.remix.com	pirulo

قد يتم استخدام أسماء مستعارة لتسمية الجهاز؛ هذا يعني أنه يمكن أن يكون هناك أسماء مختلفة لنفس عنوان IP. واجهة الإرجاع loopback نوع خاص من الواجهات التي تجعل من الممكن لعقدة الاتصال بنفسها (على سبيل المثال، للتحقق من أن النظام الفرعي للشبكة يعمل دون وصول إلى الإنترنت). بشكل مبدئي، يتم إعطاء العنوان 127,0,0,1 على وجه التحديد للإرجاع (الأمر telnet 127,0,0,1 سيتصل بنفس الجهاز). ضبط الأسماء المستعارة سهل جداً (عامة تضبطها النصوص البرمجية لتشغيل الشبكة).

مثال على loopback

```
ifconfig lo 127،0،0،1
route add host 127،0،0،1 lo
```

في الإصدار الثاني لمكتبة جنو/لينكس، هناك تغيير هام يتعلق بوظائف الملف host.conf. يشمل هذا التحسين تجميعاً مركزياً للمعلومات على الخدمات المختلفة لحلّ الأسماء، مما يعطي فوائد كثيرة لمدير الشبكة. كل المعلومات المتعلقة بمعرفة الأسماء

والخدمات قد تم تجميعها في الملف etc/nsswitch.conf/، والذي يسمح للدير بضبط الترتيب وقواعد البيانات بطريقة بسيطة جدًّا. في هذا الملف، تظهر كل الخدمات، كل خدمة في سطر، مع مجموعة من الخيارات، مثل خيار خيار حلّ اسم العقدة. يشير هذا إلى أن الترتيب في مراجعة قواعد البيانات للحصول على عنوان IP الخاص بالعقدة أو اسمها سيكون في البداية عن طريق خدمة DNS (التي تستخدم الملف etc/resolv.conf/ لتحديد عنوان IP لعقدة DNS) ومن ثم – إذا لم يمكن الحصول عليها هنا – سيتم استخدام قاعدة البيانات المحلية (في الملف etc/hosts/). من الخيارات الأخرى لهذا الغرض nis و nisplus، وهي خدمات معلومات أخرى تشرحها الوحدات التالية. الطريقة لكل مراجعة يمكن ان يتم التحكم بها عبر المهام (بين [])، على سبيل المثال:

hosts xfn nisplus dns [NOTFOUND = return] files

يشير هذا إلى أنه عند مراجعة DNS، إذا لم يكن هناك سجل لما يتم مراجعته، فالبرنامج الذي قام بالمراجعة سيعيد القيمة صفر. علامة التعجب "!" يمكن أن تستخدم لمنع العملية، على سبيل المثال:

hosts dns [!UNAVAIL = return] files

4.3 ضبط التوجيه

من النواحي الأخرى التي يجب ضبطها التوجيه. رغم أن العملية تعتبر معقدة جدًّا، إلا أنه وبشكل عام متطلبات التوجيه بسيطة جدًّا. في العقدة متعدّدة الاتصالات، يتكون التوجيه من تقرير إلى أين نرسل وماذا نستقبل. عقدة بسيطة (باتصال شبكي واحد فقط) تحتاج للتوجيه أيضاً، على اعتبار أنه لكل العقد واجهة إرجاع واتصال بالشبكة (على سبيل المثال Ethernet, PPP, SLIP, ...). كما أوضحنا سابقاً، هناك جدول يعرف بجدول التوجيه يحوي صفوفاً بحقول مختلفة، ثلاثة منها ذات أهمية خاصة: عنوان الوجهة، والواجهة التي سيتم إرسال الرسالة عبرها، وعنوان IP، والذي سيأخذ الخطوة التالية في العبّارة.

يمكن استخدام أمر التوجيه route لتغيير هذا الجدول كالقيام بمهام التوجيه الصحيحة. عندما تصل رسالة، يتم فحص عنوان الوجهة، ومقارنته بالمدخلات في الجدول، ومن ثم إرساله إلى الواجهة ذات

العنوان الأوثق ارتباطاً بوجهة الحزمة. إذا تم تحديد عبارة، فيتم إرسالها إلى الواجهة المناسبة.

لنفترض – على سبيل المثال – أن عقدتنا في شبكة من النوع C بالعنوان 192،168،110،0، وأن عنوانها 192،168،110،23، وأن الموجّه المرتبط بالإنترنت هو 192،168،110،3. سيكون الإعداد:

- أولاً، الواجهة:

```
ifconfig eth0 192،168،110،23 netmask 255،255،255،0 up
```

- ومن ثم نشير إلى أن كل البيانات للعقد التي تحمل العناوين *،192،168،0 يجب أن ترسل إلى جهاز الشبكة:

```
route add -net 192،1 ethernetmask 255،255،255،0 eth0
```

تشير net- إلى أنها مسار شبكة، ولكن يمكن أيضاً استخدام host 192،168،110،3-. سيسمح لها هذا الإعداد بأن تتصل بكل العقد ضمن جزء من الشبكة (192،1)، لكن ماذا سيحصل إذا أردنا أن نتصل بعقدة أخرى خارج هذا الجزء؟ سيكون من الصعب جدًّا الحصول على كل المدخلات المناسبة لكل الأجهزة التي نرغب بالاتصال بها. لتبسيط هذه المهمة، لدينا المسار المبدئي default route، والذي يستخدم عندما لا يطابق العنوان الهدف أيًّا من المدخلات في الجدول. من الإعدادات الممكنة لهذا:

```
route add default gw 192،168،110،3 eth0
```

(الـ gw هو عنوان IP أو اسم لعقدة موجه أو عبارة).

ومن الطرق الأخرى الممكنة لعمل ذلك:

```
ifconfig eth0 inet down
ifconfig lo
        Link encap:Local Loopback
... (لن تظهر مدخلات للواجهة eth0)
route
... (لن تظهر مدخلات في جدول التوجيه)
```

ومن ثم، نفعل الواجهة بعنوان IP جديد ومسار توجيه جديد:

```
ifconfig eth0 inet up 192،168،0،111 \
    netmask 255،255،0،0 broadcast 192،168،255،255
route add -net 10،0،0،0 netmask 255،0،0،0 \
    gw 192،168،0،1 dev eth0
```

تشير الشرطة المائلة (\) إلى أن الامر يُتبع على السطر التالي. النتيجة:

```
ifconfig
    ifconfig
    eth0 Link encap:Ethernet HWaddr 08:00:46:7A:02:B0
        inet addr:192.168.0.111 Bcast: 192.168.255.255 Mask:255.255.0.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        ...
    lo Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        ...
route
    Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192،168،0،0	*	255،255،0،0	U	0	0	0	eth0
10،0،0،0	192،168،0،1	255،0،0،0	UG	0	0	0	eth

لمزيد من المعلومات، ألق نظرة على على الأمرين ifconfig (8) و route (8).[3]

4.4 ضبط inetd

الخطوة التالية في ضبط الشبكة هي ضبط الخوادم والخدمات التي ستسمح لمستخدم آخر بالوصول إلى الجهاز المحلي أو خدماته. ستستخدم برامج الخادم المنافذ للاستماع إلى طلبات العملاء، والتي سترسل إلى هذه الخدمة على الشكل IP:port. يمكن أن تعمل الخوادم بطريقتين مختلفتين: مستقلة (تنصت الخدمة إلى المنفذ المرتبط بها وهي دائماً تعمل)، أو عبر inetd.

إن inetd خادم يتحكم باتصالات الشبكة ويديرها للخدمات المحددة في الملف /etc/inetd.conf، ويقوم

3 أظنه يقصد هنا مراجعة دليل الاستخدام بالأمر man.

بتشغيل الخادم المناسب – عندما يتم طلب خدمة – ونقل الطلب إليه.

يجب ضبط ملفين مهمين: etc/services/ و etc/inetd.conf/. في الملف الأول، نربط الخدمات، والمنافذ، والموافيق، وفي الثاني برامج الخادم التي ستردّ على طلب على منفذ معين. صيغة etc/services/ هي name port/protocol aliases، حيث الحقل الأول هو اسم الخدمة، والثاني هو المنفذ الذي تتواجد عليه الخدمة والميفاق الذي تستخدمه، والحقل الثالث هو اسم مستعار للاسم. هناك العديد من الخدمات المضبوطة مسبقاً بشكل مبدئيّ. سنعرض الآن مثالاً على etc/services/ (تشير # إلى أن ما يتبعها تعليق[4]):

tcpmux	1/tcp #	# TCP port service multiplexer
echo	7/tcp	
echo	7/udp	
discard	9/tcp	sink null
discard	9/udp sink	sink null
systat	11/tcp	users
...		
ftp	21/tcp	
ssh	22/tcp	# SSH Remote Login Protocol
ssh	22/udp	# SSH Remote Login Protocol
telnet	23/tcp	
# 24 - private		
smtp	25/tcp	mail
...		

4 أي أنه لا يقدم ولا يؤخر، وليس له أي أثر على عمل البرنامج، وإنما يوضع لمساعدة المدير على معرفة ما يحويه السطر.

4.5 ضبط inetd

الملف /etc/inetd.conf هو ملف إعداد خدمة الشبكة الرئيسية (inetd server daemon). يحوي كل سطر سبعة حقول مفصولة بفراغ: service socket_type proto flags user server_path server_args، حيث service هي الخدمة المذكورة في العمود الأول في /etc/services، و socket_type هي نوع المقبس (القيم الممكنة هي stream, dgram, raw, rdm, seqpacket)، و proto هو الميفاق الصالح لهذا الإدخال (يجب أن تطابق ذلك في /etc/services)، وتشير flags إلى ما يجب القيام به عندما يكون هناك اتصال جديدعلى خدمة تقوم باتصال آخر (wait تخبر inetd أن لا يبدأ خادماً جديداً، بينما تعني nowait أن على inetd تشغيل خادم جديد)، سيكون user اسم المستخدم المحلي الذي يعرّف به العميل الذي شغّل الخادم، و server_path هو المجلد الذي يحوي الخادم، و server_args هي معاملات ممكنة سيتم تمريرها للخادم. كمثال على بعض سطور /etc/inetd.conf (# تعليق ، فإذا كان لخدمة ما إشارة # قبل اسمها فهذا يعني أنها غير متوفرة):

```
…
telnet stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.telnetd
ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd
# fsp dgram udp wait root /usr/sbin/tcpd /usr/sbin/in.fspd
shell stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rshd
login stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rlogind
# exec stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rexecd...
…
```

في الإصدارات الحالية من دبيان تم إبدال الوظيفة inetd إلى xinetd (ينصح به)، والتي تحتاج ملف الإعداد /etc/xinetd.conf (انظر نهاية الوحدة). إذا رغبنا بتشغيل خدمة inetd، فعلينا تشغيل /etc/init.d/inetd.real start (وإنشاء الروابط المناسبة في المجلدات /etc/rcX.d) (انظر إلى نهاية هذا الجزء لأمثلة على الإعدادات). بعيداً عن إعدادات inetd و xinetd، فإن الإعداد الاعتياديّ لخدمات الشبكة لبيئة حاسوب مكتبي أو خادم بسيط قد تحوي أيضاً (بعض هذه الخدمات سيتم شرحها في الجزء المتعلق بالخوادم):

- ssh: اتصال تفاعلي آمن لاستبدال telnet وله ملفا إعداد هما /etc/ssh/ssh_config (للعميل)، و

/etc/ssh/sshd.conf (للخادم).

- Exim: خادم متعدد النقل (multi transfer agent – MTA)، ويشمل ملفات الإعداد: /etc/exim/exim.conf، و /etc/mailname، و /etc/aliases، و /etc/mail-addresses.
- Fetchmail: خادم لتنزيل البريد من حساب POP3، [وملف إعداده]: /etc/fetchmailrc.
- Procmail: برنامج لترشيح وتوزيع البريد المحلي، procmailrc./~.
- Tcpd: يرشّح الخدمات للأجهزة المفعلة والمعطلة والنطاقات للاتصال بالخادم (مغلِّفات wrappers)، [وملفات إعداده]: /etc/hosts.allow، و /etc/hosts.deny.
- DHCP: خدمة لإدارة عناوين IP (خادم) أو الحصول عليها (عميل)، /etc/dhcp3/dhclient.conf (للعميل)، و /etc/default/dhcp3-server (للخادم)، و /etc/dhcp/dhcpd.conf (للخادم).
- CVS: نظام لإدارة الإصدارات البرمجية، /etc/cvs-cron.conf، و /etc/cvs-pserver.conf.
- NFS: نظام ملفات الشبكة، /etc/exports.
- Samba: نظام ملفات شبكة ومشاركة طباعة مع شبكات وندوز، /etc/samba/smb.conf.
- Lpr: مراقب لنظام الطباعة، /etc/printcap (لنظام lpr وليس CUPS).
- Apache و Apache2: خادم وِب، /etc/apache/* و /etc/apache2/*.
- Squid: خادم وسيط و تخزين مؤقت (كاش): /etc/squid/*.

4.6 إعدادات إضافية: الموافيق والشبكات:

هناك ملفات إعداد أخرى بالكاد تُستخدَم، ولكنها قد تكون مثيرة للاهتمام. فملف etc/protocols/ يعرض محدّدات الوافيق مع أسمائها؛ بهذه الطريقة، يمكن للمبرمجين أن يحددوا الموافيق بأسمائها في البرامج.

مثال على etc/protocols/

ip	0	IP	# ميفاق الإنترنت، رقم وهمي للميفاق
#hopopt	0	HOPOPT	# خيار Hop-by-Hop لميفاق ipv6
icmp	1	ICMP	# ميفاق رسائل التحكم بالإنترنت

للملف etc/networks/ وظيفة مشابهة للملف etc/hosts/، لكن عندما يتعلق الأمر بالشبكات، فإنه يظهر أسماء الشبكات المرتبطة بعنوان IP له (سيظهر الأمر route اسم الشبكة وليس عنوانها في هذه الحالة).

مثال على etc/networks/

loopnet 127،0،0،0

localnet 192،168،0،0

amprnet 44،0،0،0 ...

4.7 النواحي الأمنية

من المهم أخذ النواحي الأمنية في اتصالات الشبكة بعين الاعتبار، حيث أن كمًّا كبيراً من الهجمات يتم عبر الشبكة. سنناقش هذا الموضوع بتعمق أكثر في الوحدة المتعلقة بالأمن؛ رغم هذا، فهناك توصيات أساسية يجب أخذها بعين الاعتبار من أجل الحد من المخاطر مباشرة قبل وبعد ضبط الشبكة في حاسوبنا.

- لا تفعل خدمات في etc/inetd.conf/ما لم تكن ستُستخدَم، أضف # قبل اسم الخدمة لتجنب مصادر الخطر.

- عدل الملف etc/ftpusers/ لمنع الوصول لمستخدمين معينين قد يكون لهم اتصال FTP بجاهزك.

- عدل الملف etc/securetty/ لتحدد من أية طرفيات (اسم لكل سطر) – على سبيل المثال: tty1, tty2, tty3, tty4 – سيكون من الممكن للمستخدم الجذر root أن يتصل. لن يتمكن المستخدم الجذر من الاتصال من أي من الطرفيات المتبقية.

- استخدم البرنامج tcpd. هذا الخادم مغلِّف (wrapper) يجعل من الممكن السماح أو منع خدمة من عقدة معينة وهو موجود في etc/inetd.conf/ كخدمة وسيطة. يتحقق tcpd من قوانين وصول معينة في ملفين، هما: etc/hosts.allow/ و etc/hosts.deny/.

إذا تم قبول الاتصال، تُبدأ الخدمة المناسبة ممررة كعامِل؛ على سبيل المثال، السطر الذي عرضناه مسبقاً لخدمة FTP في inetd.conf سيكون: ftp stream tcp nowait root /usr/sbin/tcpd/usr/in.ftpd.

يبحث tcpd أولاً في etc/hosts.allow/، ثم في etc/hosts.deny/. يحوي الملف etc/hosts.deny/القواعد التي تتعلق بالعقد التي ليس لها وصول إلى خدمة داخل هذا الجهاز. هناك إعداد مقيِّد وهو ALL:ALL والتي ستسمح بالوصول إلى الخدمات من العقد المشار إليها في etc/hosts.allow/ فقط.

يسمح الملف etc/hosts.equiv/ بالوصول إلى هذا الجهاز دون الحاجة لإدخال كلمة المرور. لا يسمح باستخدام هذه الطريقة؛ يفترض أن يُنصح المستخدمون بعدم استخدام ميزة equivalent من حساب المستخدم، وذلك عبر الملف rhosts. .

في دبيان، من المهم ضبط etc/security/access.conf/، وهو الملف الذي يحدد القوانين المتعلقة بمن يمكنه الولوج إلى هذا الجهاز ومن أين. في هذا الملف سطر بأمر له ثلاثة حقول مفصولة بالنقطتين العموديتين ":" لـ نوع التصريح: المستخدمين: المصدر. الاول سيكون + أو – (سماح أو منع)، والثاني اسم مستخدم/أسماء مستخدمين، أو مجموعة مستخدمين أو user@host، والثالث سيكون اسم الجهاز، أو العقدة، أو النطاق، أو عناوين العقدة أو الشبكات، أو كلّ ما ذكر.

مثال على access.conf

هذا الأمر لا يسمح بالولوج بالجذر عبر tty1:

ALL EXCEPT root:tty1 ...

يسمح هذا بالوصول إلى u1, u2, g1 وكل أولئك [المستخدمين والمجموعات] في النطاق remix.com. :

+:u1 u2 g1 .remox.com:ALL

4.8 خيارات ميفاق الإنترنت IP

هناك خيارات أخرى متعلقة بسير بيانات ميفاق الإنترنت علينا ذكرها. يتم ضبطها بفتح الملف المرتبط في المجلد /proc/sys/net/ipv4. اسم الملف هو نفسه كما في الأمر، ويجب وضع 1 في الملف لتفعيله، أو 0 لتعطيله.

مثال

على سبيل المثال، إذا كنا نرغب بتفعيل ip_forward، فعلينا تنفيذ:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

الأكثر استخداماً بين هذه الخيارات هي: ip_forward المستخدم للتوجيه بين الواجهات مع IP Masquerading، و ip_default_ttl وهي مدة حياة حزمة ميفاق الإنترنت (وهي 64 ميلي ثانية مبدئيًّا)، والمتغير المنطقي ip_bootp_agent الذي يقبل الحزم (أو يرفضها) بعنوان المصدر من النوع ZERO.b.c.d ووجهة هذه العقدة، broadcast أو multicast.

4.8.1 أوامر لحل مشكلات الشبكة

إذا كانت هناك مشكلات في إعداد الشبكة، يمكننا البدء بالتحقق من خرج الأوامر التالية للحصول على فكرة ابتدائية:

```
ifconfig
cat /proc/pci
cat /proc/interrupts
dmesg | more
```

من أجل التحقق من اتصال الشبكة، يمكننا استخدام الاوامر التالية (يجب أن تكون netkit-ping, traceroute, dnsutils, iptables, net-tools مثبتة):

```
ping uoc.edu                       # verifies the Internet connection
traceroute uoc.edu                 # scans IP packets
ifconfig                           # verifies the host configuration
route -n                           # verifies the routing configuration
dig [@dns.uoc.edu] www.uoc.edu     # verifies the registries in
                                   # on the dns.uoc.edu server.
iptables -L -n |less               # verifies packet filtering (kernel >=2.4)
netstat -a                         # shows all the open ports
netstat -l --inet                  # shows all the listening ports
netstat -ln --tcp                  # shoos the listening tcp ports (number)
```

5 ضبط DHCP

يرجع الاختصار DHCP إلى ميفاق الضبط الآلي للمضيف Dynamic Host Configuration Protocol. يسهل ضبط DHCP، وهو مفيد لأنه يمكن عمل هذا بطريقة مركزية بدلاً من الاضطرار لضبط كل عقدة في الشبكة على حدة، ولهذا فإدارتها أسهل. ضبط عميل سهل جداً، حيث سيكون علينا فقط تثبيت إحدى هذه الحزم: dhcp3-client (الإصدار الثالث، Internet Software Consortium)، أو dhcpcd (لكل من يويتشي هاريچوتشي و سيرجي فيزنيوك)، أو pump (لردهات)؛ وبعد ذلك نضيف dhcp إلى القسم المتعلق بالواجهة التي نرغب بأن تعمل تحت عميل dhcp (فمثلاً، etc/network/interfaces/ يجب أن تحوي iface eth0 inet dhcp ...).

يتطلب ضبط الخادم عناية أكثر، ولكنه ليس معقداً كثيراً. أولاً، ومن أجل أن يخدم الخادم كل عملاء DHCP (بما فيهم وندوز)، فعلينا أن نطرح بعض الأسئلة المتعلقة بعنوان البثّ. ولفعل هذا، أولاً يجب أن يكون الخادم قادراً على إرسال رسائل إلى العنوان 255٫255٫255٫255، وهذا غير ممكن في جنو/لينكس. لتجرب هذا نفّذ:

```
route add -host 255،255،255،255 dev eth0
```

إذا ظهرت الرسالة التالية: 255٫255٫255٫255:Unknown host، فيجب إضافة المدخلة التالية: إلى etc/hosts/: 255٫255٫255٫255 dhcp والتجربة من جديد:

```
route add -host dhcp dev eth0
```

يمكن توضيح ضبط DHCPبالواجهة الرسومية لـ linuxconf (لا ينصح به)، أو بتحرير etc/dhcpd.conf/. من الأمثلة على هذا الملف:

```
# Example of /etc/dhcpd.conf:
default-lease-time 1200;
max-lease-time 9200;
option domain-name "remix.com";
deny unknown-clients;
deny bootp;
option broadcast-address 192.168.11.255;
option routers 192.168.11.254;
option domain-name-servers 192.168.11.1,192.168.168.11.2;
subnet 192.168.11.0 netmask 255.255.255.0
{ not authoritative;
    range 192.168.11.1 192.168.11.254
    host marte {
        hardware ethernet 00:00:95:C7:06:4C;
        fixed address 192.168.11.146;
        option host-name "marte";
        }
    host saturno {
        hardware ethernet 00:00:95:C7:06:44;
        fixed address 192.168.11.147;
        option host-name "saturno";
        }
}
```

سيسمح هذا للخادم بإعطاء عناوين من النطاق 192,168,11,1 إلى 192,168,11,254، كما وصفنا لكل عقدة. إذا كان القسم { ... } المتعلق بالمضيف ذي العلاقة غير موجود، فسيتم إسنادها عشوائيًّا. يتم إسناد العناوين لوقت أقلّه 1200 ثانية وأقصاه 9200 (إذا لم تكن هذه المعاملات موجودة، فستكون غير محددة).

قبل تشغيل الخادم، علينا أن نتأكد من ما إذا كان الملف /var/state/dhcp/dhcpd.leases موجوداً (وإلا، فسيكون علينا إنشاؤه بالأمر touch /var/state/dhcp/dhcpd.leases). لتشغيل الخادم: /usr/sbin/dhcp (أو يمكننا وضعه في نصوص بدء التشغيل). يمكننا بالأمر /usr/sbin/dhcpd -d -f أن نرى النشاط في الخادم في الواجهة النصيّة للنظام.

علينا أن لا ننسى العِبارة not authoritative، حيث أنه إذا حصل ذلك [وتجاهلنا الأمر]، فقد يترك هذا الخادمُ

خادمَ dhcp آخر يخدم عناوين الإنترنت لأجزاء أخرى غير مفعّل.

6 عناوين IP المستعارة

هناك بعض التطبيقات التي يكون من المفيد فيها ضبط العديد من عناوين الإنترنت لجهاز شبكة واحد. يستخدم مزودو خدمة الإنترنت ISPs هذه الخاصية كثيراً لتقديم مزايا مخصصة (كالشبكة العنكبوتية العالمية و FTP) لمستخدميها. ولهذا، يجب أن تكون النواة مصرّفة مع خيارات Network Aliasing و ودعم IP aliasing. بعد تثبيت النواة الجديدة، فالضبط سهل جدًّا. ترتبط العناوين المستعارة بأجهزة الشبكة الافتراضية المرتبطة بالجهاز الجديد بصيغة تشبه اسم الجهاز تليها نقطتان عموديتان ثم رقم افتراضي.

على سبيل المثال، eth0:0 و ppp0:8

لنقل بأن لدينا شبكة إثرنت تدعم شبكتين فرعيتين مختلفتين في نفس الوقت، وأننا نريد لجهازنا أن يكون له اتصال مباشر بهما. من الأمثلة على هذا الإعداد:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
route add -net 192.168.110.0 netmask 255.255.255.0 eth0
ifconfig eth0:0 192.168.10.23 netmask 255.255.255.0 up
route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

مما يعني أنه سيكون لدينا عنوانا إنترنت، وهما 192,168,110,23 و 192,168,10,23 لنفس بطاقة الشبكة. لحذف عنوان مستعار، أضف – إلى نهاية الاسم (على سبيل المثال، ifconfig eth0:0- 0).

وهناك حالة اعتيادية وهي عندما نرغب بضبط بطاقة شبكة واحدة لتعمل وكأنها واجهة لشبكة فرعية مختلفة. على سبيل المثال، لنفرض بأن لدينا جهاز على شبكة LAN، وهي LAN 192,168,0,x/24، وأننا نرغب بوصل الجهاز بالإنترنت

باستخدام عنوان IP عام مقدم من DHCP باستخدام بطاقة الشبكة الموجودة. على سبيل المثال، يمكننا اتباع الإجراءات الموصوفة في المثال السابق أو تحرير الملف etc/network/interfaces/ وبهذا تشمل جزءاً مشابهاً لما يلي:

```
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0
network 192.168.0.0
```

```
broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

الواجهة eth0:0 هي واجهة افتراضية وسُتفعّل الواجهة الأم لها – أي eth0 – عندما تُفعّل.

7 IP masquerade

إن IP Masquerade مورد يستخدم بحيث تتمكن مجموعة أجهزة من استخدام عنوان واحد. يسمح هذا للعقد المخفية (وبمعنى آخر، تلك التي تستخدم عنواناً خاصًّا، مثل 192,168,10,1) بالوصول إلى الإنترنت، لكن لا يمكنها قبول الخدمات أو النداءات الخارجية مباشرة، بل فقط عبر الجهاز الذي يملك عنوان IP الحقيقي.

يعني هذا بأن بعض الخدمات لن تعمل (على سبيل المثال، talk)، وأن بعضها الآخر يجب أن يضبط في الوضع PASV (أي passive) لكي تعمل (مثل FTP). لكن الوِب و telnet و IRC ستعمل جيداً. يجب أن تضبط النواة بالخيارات التالية: جدران حماية الشبكة، وشبكات TCP/IP، و IP: forwarding/gatewaying، و IP: Masquerading. في العادة، الإعداد الأكثر شيوعاً هو أن يكون لدينا جهاز باتصال SLIP أو PPP وأن يكون به جهاز شبكة آخر (على سبيل المثال، بطاقة إثرنت) وعنوان شبكة محجوز. كما رأينا، وكما ذُكِر في RFC 1918، نطاق العناوين التالي (IP/Mask) يمكن استخدامه كنطاق عناوين خاصة: 10,0,0,0/255,0,0,0، و 172,16,0,0/255,240,0,0، و 192,168,0,0/255,255,0,0. العقد التي يجب عمل masquerade لها ستكون على هذه الشبكة الثانية. يجب أن يكون لكلّ من هذه الأجهزة عنوان الجهاز الذي يقوم بالعملية كعبّارة مبدئية أو موجه. في هذا الجهاز، يمكننا أن نضبط:

- مسار شبكة لإثرنت تعتبر أن للشبكة عنوان IP وهو 192,168,1,0/255,255,255,0:

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

- مسار مبدئي لبقية الإنترنت:

```
route add default ppp0
```

- كل العقد على الشبكة 192,168,1/24 سيتم عمل masqerade لها:

```
ipchains -A forward -s 192.168.1.0/24 -j MASQ
```

- إذا كان iptables مستخدماً في النواة، الإصدار 2,4 والأحدث:

```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

راجع المراجع في الوحدة التي تغطي الأمن لمعلومات عن ipchains و iptables.

8 NAT في النواة 2,2 والأحدث

ترجمة عناوين الشبكة NAT هي بديل قد أعطى ميزة IP Masquerade الأثرية لجنو/لينكس، والتي تقدم مزايا إضافية للخدمة. إن إحدى التحسينات المضمنة في مجموعة TCP/IP لجنو/لينكس 2,2 وهي NAT مضمنة في النواة. ولاستخدامها، علينا تصريف النواة بـ: CONFIG_IP_ADVANCED_ROUTER، و CONFIG_IP_MULTIPLE_TABLES، و CONFIG_IP_ROUTE_NAT.

وإذا كنا نحتاج تحكماً منطقياً بقوانين NAT (على سبيل المثال، لتفعيل جدار النار)، فيجب أن يكون لدينا: CONFIG_IP_FIREWALL و CONFIG_IP_ROUTE_FWMARK.

من أجل العمل بهذه المزايا الجديدة، نحتاج لأن نستخدم البرنامج ip (والذي يمكن الحصول عليه من ftp://ftp.inr.ac.ru/ip_routing/). ومن ثم، لترجمة عناوين البيانات القادمة، يمكننا استخدام:

```
ip add route nat <extaddr>[/<masklen>] via <intaddr>
```

سيترجم هذا عنوان الوجهة للحزم القادمة المعنونة إلى ext-addr (العنوان المرئي خارجياً على الإنترنت) إلى int-addr (عنوان الشبكة الداخلية عبر العبّارة أو جدار النار). يتم توجيه الحزمة بما يتناسب مع جدول التوجيه المحلي. يمكن ترجمة العناوين المفردة أو كتل العناوين. على سبيل المثال:

```
ip route add nat 240،0،11،34 via 192،109،0،2
ip route add nat 240،0،11،32/27 via 192،109،0،0
```

يجعل السطر الأول الوصول للعنوان الداخلي 192,109,0,2 ممكناً عبر العنوان 240,0,11,34. أما الثاني فيعيد تعريف كتلة العناوين 192,109,0,0/31 إلى 240,0,11,32/63. في هذه الحالة، لقد استخدمنا كمثال ترجمة إلى عناوين من المجموعة D و E، مثل *,*,240,0 وذلك لكي لا تستخدم عنواناً عامًّا. يجب أن يغير المستخدم هذه العناوين (240,11,0,34 و 240,0,11,32/63) إلى العناوين العامة ذات العلاقة التي يرغب بأن يترجمها.

9 كيفيّة ضبط اتصالات DialUP و PPP

إن ضبط اتصال طلب هاتفي باستخدام PPP في جنو/لينكس سهل جدًّا. يجعل PPP (ميفاق الاتصال من نقطة إلى نقطة) من الممكن إنشاء روابط عناوين إنترنت بين حاسوبين لكل منهما مودم (يجب أن يكون مودماً يدعمه جنو/لينكس، حيث أنه ليس كل المودمات – خاصة المودمات الداخلية أو تلك المعروفة بـ winmodems – يمكن ضبطها، وذلك لأن كثيراً منها تحتاج برمجيات إضافية من أجل إنشاء اتصال).

للبدء يجب أن يكون لدينا المعلومات التالية: الـ init-string الخاص بالمودم (هذا ليس ضرورياً في العادة، لكن إن كان ضرورياً ولم يكن متوفراً، فيمكننا استخدام ATZ الذي يعمل في معظم المودمات، أو يمكننا مراجعة قوائم init-strings متخصصة).

نحتاج أيضاً لمعلومات مزود خدمة الإنترنت ISP: هوية الاتصال (اسم الولوج)، وكلمة المرور، ورقم الهاتف. ينصح أيضاً بأن يكون لدينا عناوين DNS، لكن هذا اختياري في الإصدارات الحالية من pppd. علينا أيضاً أن نتحقق من أن المودم متصل بشكل جيد. في حالة المودمات الخارجية، علينا تنفيذ echo > /dev/ttyS وتفحص الثنائيات المضيئة في المودم لنرى إن كانت تضيء. إن لم تفعل، جرب مع ttyS1، إذا كان المودم متصلاً بالمنفذ التسلسلي الثاني. في حالة المودم الداخلي، تفحص دليل العتاد المدعوم لترى إذا كان يمكن أن يتعرف جنو/لينكس على هذا المودم؛ إذا كان هذا صحيحاً، فقد يكون من الضروري إعادة ضبط النواة من أجل استخدامه. يمكننا أيضاً استخدام cat /proc/pci في حال كان متصلاً عبر منفذ PCI.

إن أسهل طريقة لضبط المودم الآن هي عبر حزمة kppp (يجب أن نثبت الحزم *kde-network-ppp و *ppp). نفّذ في الطرفية usr/bin/kppp/. في النافذة، املأ الصناديق كما يلي:

Accounts ⇒ الاتصال الجديد
Dial ⇒ الاستيثاق 'PAP/CHAP'
Store Password ⇒ yes
IP ⇒ Dynamic IP Address
Autoconfigure hostname ⇒ No

Gateway ⇒المبدئي

Gateway ⇒ Assign the Default Route
DNS ⇒ Configuration Automatic ⇒Disable existing DNS
Device ⇒ ttyS1(com1) o ttyS2 (com2)
Modem ⇒ استعلم عن المودم لترى النتائج (إذا لم تحصل على نتائج، فعدّل الجهاز ttySx)

بعد إدخال اسم الولوج وكلمة المرور، سيتم وصلنا بالإنترنت (للتأكد من أننا متصلون، نفّذ ping www.google.com على سبيل المثال). لقد استخدمنا هنا الحزمة kppp، لكن كان بإمكاننا استخدام linuxconf و gnomeppp.

هناك طريقة سريعة لضبط pppd في ديبان وتعتمد على استخدام البرنامج pppconfig، والذي يأتي مع الحزمة. يضبط pppconfig ملفات مثل الملفات السابقة بعد أن يأتي مع حزمة wvdial. بدلاً من جعل pppd ينشيء محادثة لطلب والمفاوضة على الاتصال، يقوم wvdial بالطلب والتفاوض الابتدائي، ومن ثم يبدأ pppd بحيث يتمكن من عمل الباقي. في معظم الحالات، يمكن لـ wvdial بدء الاتصال باستخدام رقم الهاتف واسم المستخدم وكلمة المرور فقط.

بمجرد ضبط PPP لكي يعمل مع my_isp – على سبيل المثال – ، يجب علينا أن نحرر الملف /etc/network/interfaces بحيث تتضمن قسماً كالتالي (الأمران ifup و ifdown يستخدمان الأمرين pon و poff لضبط واجهات ppp):

```
iface ppp0 inet ppp
provider my_isp
```

في هذا القسم، ifup ppp0 ينفّذ:

```
pon my_isp
```

ليس من الممكن حالياً استخدام ifup لعمل إعداد مساعد لواجهات PPP. وحيث يختفي pon قبل أن ينتهي pppd من إنشاء الاتصال، ينفذ ifup النصوص البرمجية لتشغيل الواجهة قبل أن تكون واجهة PPP جاهزة للاستخدام. إلى أن يتم حل هذا الخطأ، فسيبقى من الضروري ضبط الاتصال لاحقاً في /etc/ppp/ip-up أو /etc/ppp/ip-up.d/.

يستخدم العديد من مزودي خدمات الإنترنت عريض النطاق PPP للتفاوض على الاتصال حتى عندما تكون أجهزة

العملاء متصلة عبر شبكات ATM و/أو إثرنت. يتم هذا عبر الميفاق PPP over Ethernet – PPPoE، وهي تقنية لتضمين سير PPP عبر إطارات إثرنت. لنفترض أن اسم مزود الخدمة هو my_ISP. أولاً، يجب علينا أن نضبط PPP و PPPoE لـ my_isp. تعتمد الطريقة الأسهل في عمل هذا على تثبيت الحزمة pppoeconf وتنفيذ pppoeconf في الواجهة النصية. ومن ثم نحرر الملف etc/network/interfaces/ بحيث يتضمن جزءاً يشبه ما يلي:

```
iface eth0 inet ppp
provider my_isp
```

في بعض الأحيان تظهر مشكلات في PPPoE تتعلق بالحدّ الأقصى لوحدة النقل (maximum transmit unit أو MTU) في خطوط DSL؛ يمكنك مراجعة DSL-HOWTO لمزيد من التفاصيل. إذا كان في المودم موجّه، بحيث يتعامل المودم/الموجه مع اتصال PPPoE بنفسها، وستظهر في جهة الشبكة المحلية كعبّارة من إثرنت إلى الإنترنت.

10 ضبط الشبكة عبر hotplug

تدعم الحزمة hotplug الإبدال الحيّ عند الإقلاع (يجب ان تكون الحزمة المعنية مثبتة). يمكن إضافة عتاد الشبكة إما عند بدء التشغيل، أو بعد إضافة البطاقة إلى الجهاز (بطاقة PCMCIA على سبيل المثال)، أو بعد تشغيل أداة مثل discover وتحميل الوحدات الضرورية. عندما تكتشف النواة عتاداً جديداً، فإنها تشغّل مشغل العتاد ومن ثم برنامج hotplug لضبطه. إذا تم إزالة الجهاز فيما بعد، يشغّل البرنامجُ hotplug مرة أخرى، بمعاملات مختلفة. في دبيان، عند طلب hotplug، فهذا ينفذ النصوص البرمجية في /etc/hotplug/ و /etc/hotplug.d/. عتاد الشبكة الذي تم وصله مؤخراً يتم ضبطه عبر /etc/hotplug/net.agent. لنفترض أنه تم توصيل بطاقة شبكة PCMCIA، مما يعني أن الواجهة eth0 ستكون جاهزة للاستخدام. ينفذ /etc/hotplug/net.agent التالي:

```
ifup eth0=hotplug
```

ما لم تكن هناك واجهة منطقية بالاسم hotplug تم إضافتها إلى /etc/network/interfaces/، فلن يكون لهذا الأمر أي أثر. لكي يضبط هذا الأمر eth0، علينا إضافة السطور التالية إلى /etc/network/interfaces/:

```
mapping hotplug
script echo
```

إذا كنت تريد عمل hotplug لـ eth0 فقط، وليس لأي واجهة أخرى، استخدم grep بدلاً من echo كما يلي:

```
mapping hotplug
script grep
map eth0
```

ifplugd تفعل وتعطل الواجهة اعتماداً على ما إذا كان العتاد الذي يندرج تحتها متصلاً بالشبكة أم لا. يمكن للبرنامج اكتشاف اتصال سلك بواجهة إثرنت، أو نقطة وصول متصلة بواجهة Wi-Fi. عندما يرى ifplugd أن حالة الاتصال تغيرت، فسينفذ نصاً برمجياً يقوم – مبدئيًّا – بتنفيذ ifup أو ifdown للواجهة. يعمل ifplugd جماعيًّا مع hotplug. عند إضافة بطاقة، مما يعني أن الواجهة جاهزة للاستخدام، يشغّل /etc/hotplug.d/net/ifplugd.hotplug نسخة عن ifplug لتلك الواجهة. عندما

يكتشف ifplugd أن البطاقة متصلة بشبكة، فإنها تنفّذ ifup لهذه الواجهة.

من أجل ربط واجهة Wi-Fi بنقطة وصول، فقد يكون علينا برمجتها مع كود تشفير WEP مناسب. إذا كان ifplugd مستخدماً للتحكم بـ ifup – كما شرحنا –، فمن الوضح أنه لن يكون قادراً على ضبط كود التشفير باستخدام ifup، حيث أنه يستدعى فقط عندما يتم ربط البطاقة [بالشبكة اللاسلكية]. أبسط الحلول استخدام waproamd الذي يضبط كود تشفير WEP اعتماداً على نقاط الوصول المتاحة المكتشفة عبر البحث في الشبكات اللاسلكية. للمزيد من المعلومات، راجع man waproamd والمعلومات عن الحزمة.

11 الشبكات الافتراضية الخاصة VPN

إن الشبكة الافتراضية الخاصة Virtual Private Network – VPN هي شبكة تستخدم الإنترنت لنقل البيانات، ولكنها تمنع المستخدمين الخارجيين من الوصول إلى البيانات.[5]

هذا يعني أن لدينا شبكة بعقد VPN متصلة و [تلك الشبكة] ممررة عبر شبكة أخرى تمر عبرها البيانات ولا يمكن لأحد منها التفاعل مع تلك الشبكة [الممررة (شبكة VPN) من خارجها]. وتستخدم عندما يرغب مستخدمون بعيدون بالوصول إلى شبكة شركة للحفاظ على أمن وخصوصية البيانات. هناك العديد من الطرق التي يمكن استخدامها لضبط VPN، مثل SSH (SSL), CIPE, IPSec, PPTP، يمكن الرجوع إليها في المراجع (ننصح بمراجعة VPN PPP-SSH HOWTO لسكوت برونسون، و VPN-HOWTO لماثيو د. ويلسون).

من أجل القيام باختبارات الإعداد في هذا القسم، سنستخدم OpenVPN، وهو حل معتمد على SSL VPN ويمكن أن يستخدم لنطاق عريض من الحلول؛ على سبيل المثال، الوصول عن بعد، أو VPN نقطة إلى نقطة، أو شبكات WiFi آمنة، أو شبكات الشركات الموزعة. يستخدم OpenVPN الطبقة الثانية أو الثالثة من OSI باستخدام الموافيق SSL/TSL ويدعم الاستيثاق المعتمد على الشهادات والبطاقات الذكية وطرق التأكيد الأخرى. OpenVPN ليس خادم تطبيقات الوسيط ولا يعمل عبر متصفح إنترنت.

من أجل تحليله، سنستخدم خياراً في OpenVPN يدعى OpenVPN لإعدادات المفتاح الثابت، والتي توفر طريقة بسيطة لضبط VPN مثالي للاختبارات أو للاتصالات من نقطة إلى نقطة. مزاياها هي البساطة، وحقيقة أنه ليس من الضروري أن يكون لدينا شهادة البنية التحتية للمفتاح العام (Public Key Infrastructure أو PKI اختصاراً) X509 للحفاظ على الـ VPN. العيوب هو كونها تسمح فقط لعميل واحد وخادم واحد، وذلك لأنه بسبب عدم استخدام المفتاحين العام والخاص، قد تكون لدينا نفس المفاتيح من جلسات سابقة، ويجب أن يكون هناك وضع نصيّ في كل نظير، وأن يكون المفتاح السري قد تم تبادله للحصول على

---

5 للمزيد، راجع ويكيبيديا، الرابط: https://en.wikipedia.org/wiki/VPN.

قناة آمنة.

11.1. مثال بسيط

في هذا المثال، سنضبط قناة VPN على خادم بالعنوان 10,8,0,1 وعميل بالعنوان 10,8,0,2. الاتصال سيكون مشفراً بين العميل والخادم على منفذ UDP ذي الرقم 1194، وهو المنفذ المبدئي في OpenVPN. بعد تثبيت الحزمة (http://openvpn.net/install.html)، يجب علينا إنشاء المفتاح الثابت:

```
openvpn --genkey --secret static.key
```

ثم يجب علينا نسخ الملف static.key إلى النظير الآخر عبر قناة آمنة (باستخدام ssh أو scp على سبيل المثال). ملف إعداد الخادم openVPN_server على سبيل المثال:

```
dev tun
ifconfig 10،8،0،1 10،8،0،2
secret static.key
```

ملف إعداد العميل، على سبيل المثال openVPN_client

```
remote myremote.mydomain
dev tun
ifconfig 10،8،0،2 10،8،0،1
secret static.key
```

قبل التحقق من أن VPN يعمل، علينا التحقق من جدار الحماية من أن المنفذ UDP 1194 مفتوح على الخادم، وأن الواجهة الافتراضية tun0 المستخدمة من OpenVPN غير محظورة، سواء عند العميل أو عند الخادم. أبق في بالك أن 90% من مشاكل الاتصال التي يواجهها مستخدمو OpenVPN الجدد تتعلق بطريقة ما بجدار الحماية.

من أجل التحقق من OpenVPN بين الجهازين، يجب علينا أن نغير العناوين إلى العنوانين الحقيقيين والنطاق إلى النطاق ذي العلاقة، ومن ثم تنفيذ [التالي] من جهة الخادم:

```
openvpn [server config file]
```

الذي سيقدم مخرجات مثل:

```
Sun Feb 6 20:46:38 2005 OpenVPN 2.0_rc12 i686-suse-linux [SSL] [LZO] [EPOLL] built on Feb 5 2005
Sun Feb 6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key
Sun Feb 6 20:46:38 2005 TLS-Auth MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Sun Feb 6 20:46:38 2005 TUN/TAP device tun1 opened
Sun Feb 6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2 mtu 1500
Sun Feb 6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask 255.255.255.0 gw 10.8.0.2
Sun Feb 6 20:46:38 2005 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:23 ET:0 EL:0 AF:3/1 ]
Sun Feb 6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194
Sun Feb 6 20:46:38 2005 UDPv4 link remote: [undef]
Sun Feb 6 20:46:38 2005 MULTI: multi_init called, r=256 v=256
Sun Feb 6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62
Sun Feb 6 20:46:38 2005 IFCONFIG POOL LIST
Sun Feb 6 20:46:38 2005 Initialization Sequence Completed
```

وفي جهة العميل:

```
openvpn [client config file]
```

من أجل تفحص ما إذا كانت تعمل، يمكننا تنفيذ ping 10,8,0,2 من الخادم، أو ping 10,8,0,1 من العميل. لمزيد من المعلومات، تفحص http://openvpn.net/howto.html.

لإضافة الضغط إلى الاتصال، يجب علينا إضافة السطر التالي إلى ملفي الإعدادات:

```
comp-lzo
```

من أجل حماية الاتصال وإبقائه عبر موجهات وجدران حماية NAT وإكمال تغييرات IP عبر DNS، فإذا تغير أحد النظيرين، أضف التالي إلى ملفي الإعداد:

```
keng-timer-rem
persist-tun
peepalive 10 60
pirsist-key
```

للتشغيل كمراقب بصلاحيات المستخدم والمجموعة nobody، أضف التالي إلى ملف الإعدادات:

user nobody

group nobody

Daemon

12 أدوات وإعدادات متقدمة

هناك مجموعة من الحزم الإضافية (التي تستبدل الحزم المعهودة) والأدوات التي إما تحسن أمن الجهاز (مستحسنة في البيئات العدائية) أو للمساعدة على ضبط الشبكة (والنظام بشكل عام) بأسلوب أكثر قرباً من المستخدم.

> قد تكون هذه الحزم مفيدة جدًّا لمدير الشبكة لتجنب الاختراقات، أو لتجنب تجاوز مستخدمين محليين صلاحياتهم (والتي عادة لا ينفذها المستخدمون المحليون، بل شخص ما ينتحل شخصيتهم) أو لمساعدة المستخدمين الجدد على ضبط الخدمات بشكل مناسب.

ولهذا الغرض، يجب علينا أن نختبر:

- إعداد TCP/IP متقدم: يمكن استخدام الأمر sysctl لتعديل معاملات النواة أثناء التشغيل أو عند الإقلاع، لضبطها لتناسب احتياجات النظام. المعاملات التي يمكن تعديلها هي تلك الموجودة في المجلد /proc/sys/ والتي يمكن مراجعتها بالأمر sysctl -a. الطريقة الأسهل لتعديل هذه المعاملات هي عبر ملف الإعداد /etc/sysctl.conf. بعد القيام بالتعديلات، يجب علينا أن نعيد تشغيل الشبكة:

```
/etc/init.d/networking restart
```

في هذا القسم، سنختبر بعض التعديلات لتحسين أداء الشبكة (تحسينات معتمدة على الأوضاع) أو على أمن النظام (تفقد المراجع لمزيد من المعلومات):

```
net.ipv4.icmp_echo_ignore_all = 1
```

- لا تستجب لحزم ICMP، كالأمر ping على سبيل المثال، مما قد يعني أن هناك هجمة حجب خدمة DoS.

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

- تجنب الاكتظاظ في الشبكة التي لا تستجيب إلى broadcast.

```
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.lo.accept_source_route = 0
```

```
net.ipv4.conf.eth0.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
```

- امنع حزم توجيه مصدر IP، التي قد تشكل تهديداً أمنياً (في كلّ الواجهات).

```
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.accept_redirects = 0
```

- اسمح لرفض هجمات حجب الخدمة DoS عبر حزم SYNC، والتي يمكن أن تستنزف كل موارد النظام، مضطرة المستخدم لإعادة تشغيل الجهاز.

```
net.ipv4.conf.lo.accept_redirects = 0
net.ipv4.conf.eth0.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
```

- مفيد لتجنب هجمات قبول إعادة توجيه ICMP (تستخدم هذه الحزم عندما لا يكون للتوجيه وجهة مناسبة) في كل الواجهات.

```
net.ipv4.icmp_ignore_bogus_error_responses = 0
```

- أرسل تنبيهات عن كل رسائل الخطأ في الشبكة.

```
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.lo.rp_filter = 1
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
```

- تفعل الحماية ضد انتحال عناوين IP في كل الواجهات:

```
net.ipv4.tcp_fin_timeout = 40, By default, 60.
net.ipv4.tcp_keepalive_time = 3600, By default, 7.200.
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_sack = 0
net.ipv4.tcp_timestamps = 0, By default, all at 1 (enabled).
```

- Iptables: الإصدار الأحدث من جنو/لينكس (النواة 2,4 والأحدث) يتضمن ميزة بناء مرشحات حزم تسمى netfilter. يتحكم بهذه الميزة أداة يطلق عليها iptables التي لها خصائص أفضل من سلفها (ipchains). كما سنرى في الوحدة المتعلقة بالأمن، من السهل جدًّا بناء جدار حماية بهذه الأداة لاكتشاف وصدّ الهجمات الأكثر شيوعاً،

مثل DoS، و انتحال IP و MAC، إلخ. قبل تفعيلها، علينا التأكد من أن النواة بالإصدار 2,4 أو الأحدث، وهي الإصدار المضبوط لدعم ipfilters (مما يعني أنه من الضروريّ تصريف النواة لتفعيل الخيار "ترشيح حزم الشبكة" network packet filtering – أي [CONFIG_NETFILTER] – وكل الخيارات الفرعية المحددة). هناك قواعد محدّدة يجب أن تكون مفعلة عند الإقلاع (على سبيل المثال، عبر /etc/init.d/ والرابط المناسب في مجلد rc المناسب) وسيكون لدينا صيغة مشابهة لما يلي[6] (تفقّد المراجع لمعرفة الإمكانيات والصياغة الكاملة):

```
iptables -A Type -i Interface -p protocol -s SourceIP --source-port Port -d
DestinationIP --destination-port Port -j Action
```

- GnuPG: هذه الأداة تجعل بالإمكان تعمية البيانات ليتم إرسالها (بالبريد مثلاً) أو تخزينها فيما بعد، ويمكنها أيضاً عمل تواقيع رقمية (توافق معيار RFC2440) ولا تستخدم خوارزميات عليها براءات اختراع، مما يعني أنها مفتوحة المصدر، ولكننا نفقد التوافقية مع أدوات أخرى (مثل PGP 2,0) تستخدم خوارزميات مثل IDEA و RSA. لتصريف و/أو تثبيت الأداة، اتبع تعليمات المبرمجين في http://www.gnupg.org[7]. بداية، علينا أن ننشئ زوجاً من المفاتيح (عام وخاص) بتنفيذ الأمر gpg --gen-key – بصلاحيات الجذر – مرتين والإجابة على الأسئلة التي تظهر. عامة، ستخزن هذه المفاتيح في /root/. ثم نصدر المفتاح العام (إلى موقع مثلاً) بحيث يتمكن المستخدمون الآخرون من استخدامه لتعمية البريد/المعلومات مما يجعل من الممكن رؤيتها[8] فقط للمستخدم الذي أنشأ المفتاح العام. لعمل ذلك، علينا أن نستخدم الأمر gpg --export -ao UID الذي سينشئ ملف آسكي للمفتاح العام للمستخدم ذي المعرف UID. من أجل استيراد المفتاح العام لمستخدم آخر، يمكننا استخدام الأمر gpg --import متبوعاً باسم الملف، ولتوقيع مفتاح (أي أن نقول

6 وتكتب على سطر واحد، وإنما وضعت على سطرين هنا لقلة عرض الصفحة. يمكن تقسيم الأمر إلى أكثر من سطر باستخدام الشرطة المائلة "\" في نهاية كل سطر عدا الأخير.

7 تذكر أن أبسط وأسلم الطرق للتثبيت هي استخدام المستودعات الرسمية للتوزيعة. أداة GnuPG متوفرة في مستودعات معظم التوزيعات الحديثة المشهورة، وقد تكون مثبتة مسبقاً لديك، لذا تأكد من الأمر.

8 ويقصد برؤيتها هنا أي معرفة محتواها.

للنظام بأننا واثقون من أن المفتاح الموقع هو فعلاً ممن يشير إلى أنه منه)، فيمكننا استخدام الأمر gpg --sign-key UID. للتحقق من مفتاح، يمكننا استخدام gpg --verify file/data، ولتعمية/إلغاء تعمية مفتاح، نستخدم gpg -sear file- و gpg -d file على التوالي.

- Logcheck: إن إحدى المهام الرئيسية لمدير الشبكة تفقد ملفات التقارير يوميًّا (أكثر من مرة في اليوم) لاكتشاف أية هجمات/اختراقات أو أحداث محتملة حيث يمكن أن تكون دليلاً عليها. تختار هذه الاداة معلومات مركّزة حول المشكلات والمخاطر المحتملة (من ملفات التقارير)، ومن ثم ترسل هذه المعلومات إلى المدير ذي العلاقة – عبر البريد على سبيل المثال –. تتضمن الحزمة أدوات لتنفيذ نمط مستقل وتذكر آخر مدخلة تم التحقق منها للتشغيل التالي. لمعلومات عن الإعداد/التثبيت، يمكنك مراجعة المراجع.

- PortSentry و TripWire: تساعد هذه الادوات مدير الشبكة للقيام بمهامه الأمنية. يجعل PortSentry من الممكن اكتشاف والاستجابة لعمليات البحث في المنافذ (الخطوة المسبقة قبل الهجوم أو السُخام[9]) في الوقت الحقيقي، وعمل قرارات عديدة متعلقة بالأفعال التي تتم في تلك اللحظة. ستساعد أداة TripWire المدراء بتنبيههم عند حصول تعديلات أو تغييرات في الملفات، لتجنب أي تلف (خطير) محتمل. تقارن هذه الأداة الاختلافات بين الملفات الحالية وقاعدة بيانات أُنشئت مسبقاً لاكتشاف التغييرات (إدخالاً و حذفاً)، وهذا مفيد جدًّا لاكتشاف التغييرات المحتملة للملفات الحيوية كملفات الإعداد مثلاً. عد إلى المراجع لمعلومات عن تثبيت/ضبط هذه الأدوات.

- Xinetd: تحسن هذه الاداة الكفاءة والأداة بشكل كبير لكل من inetd و tcp-wrappers. من أهم مزايا xinetd هو أنه بإمكانها تفادي هجمات حجب الخدمة DoS عبر آليات التحكم للخدمات المعتمدة على معرفة عنوان العميل خلال وقت الوصول ووقت (الولوج). يجب أن لا يفترض بأن Xinetd هو أنسب خيار لكل الخدمات (على سبيل المثال، من الأفضل أن يعمل FTP و SSH كمراقبين فقط)، حيث أن أياً من هذه الخدمات ستتسبب بحمل زائد على النظام وهناك آليات للوصول الآمن لا تسبب انقطاعاً في أمن النظام.

---

9 السُخام (بالإنجليزية spam)، هي الرسائل المزعجة أو غير المرغوب بها، كالإعلانات التطفلية مثلاً.

التصريف و/أو التثبيت بسيط، علينا فقط ان نضبط ملفين: /etc/xinetd.conf (ملف إعداد Xinetd)، و /etc/rc.d/init.d/xinet.d (ملف تشغيل xinetd). الملف الأول يحوي قسمين: defaults، وهو المكان الذي سنجد فيه المعاملات التي تنطبق على كل الخدمات، وقسم الخدمات services، وسيحوي الخدمات التي يشغلها Xinetd.

فيما يلي مثال اعتيادي على الإعداد:

```
# xinetd.conf

# The default configuration options that are applied to all the

# servers may be modified for each service
defaults
{
instances = 10
log_type = FILE /var/log/service.log
log_on_success = HOST PID
log_on_failure = HOST RECORD
}

# The name of the service must be located in /etc/services to obtain

# the right port

# If the server/Port is not a standard one, use "port = X"

service ftp
{
socket_type = stream
protocol = tcp
wait = no
user = root
server = /usr/sbin/proftpd
}
#service telnet
#{
# socket_type = stream
# protocol = tcp
# wait = no
# user = root
# no_access = 0.0.0.0
# only_from = 127.0.0.1
# banner_fail = /etc/telnet_fail
# server = /usr/sbin/in.telnetd
#}
service ssh
{
```

```
socket_type = stream
protocol = tcp
wait = no
user = root
port = 22
server = /usr/sbin/sshd
server_args = -i
}
service http
{
socket_type = stream
protocol = tcp
wait = no
user = root
server = /usr/local/apache/bin/httpd
}
#service finger
#{
# socket_type = stream
# protocol = tcp
# wait = no
# user = root
# no_access = 0.0.0.0
# only_from = 127.0.0.1
# banner_fail = /etc/finger_fail
# server = /usr/sbin/in.fingerd
# server_args = -l
#}
# End of /etc/xinetd.conf
```

الخدمات المذكورة أعلاه (والمبدوءة بالرمز #) لن تكون متاحة. في القسم defaults يمكننا تثبيت معاملات مثل الحدّ الأقصى لعدد الطلبات المتزامنة للخدمة، ونوع السجلّ (log) الذي نطلبه، من أي عقد ستُستقبَل الطلبات مبدئيًّا، ووالعدد الأقصى لطلبات IP التي سيتم الاستماع إليها، والخدمات التي يتم تشغيلها كخوادم superservers (مثل imapd أو popd)، مثل:

```
default {
instances = 20
log_type = SYSLOG
authpriv log_on_success = HOST
log_on_failure = HOST
only_from = 192.168.0.0/16
per_source = 3
enabled = imaps
}
```

قسم الخدمات - واحد لكل خدمة - مثل:

```
service imapd {
socket_type = stream
wait = no
user = root
server = /usr/sbin/imapd
only_from = 0.0.0.0/0 #allows every client
no_access = 192.168.0.1
instances = 30
log_on_success += DURATION USERID
log_on_failure += USERID
nice = 2

redirect = 192.168.1.1 993 #Makes it possible to redirect the traffic of port 993

to node 192.168.1.1
bind = 192.168.10.4

#Makes it possible to indicate the interface to which the service is associated to
avoid service spoofing problems.

}
```

يجعل الملف etc/init.d/xinetd/ بدء الخادم (مع الرابط المناسب - اعتماداً على مستوى التشغيل المختار -، على سبيل المثال، 3 و 4 و 5) ممكناً. من المفيد تغيير الخصائص لكلي الملفين لضمان أن لا يتم تعديلهما أو تعطيلهما وذلك بتنفيذ:

```
chmod 700 /etc/init.d/xinetd; chown 0.0 /etc/init.d/xconfig; \
```

```
chmod 400 /etc/xinetd.conf; chattr +i /etc/xinetd.conf
```

- Linuxconf: هذه أداة إعداد وإدارة لأنظمة جنو/لينكس، ولكنها تعد أثرية لمعظم التوزيعات الحالية، رغم هذا فمن الممكن أن توجد في بعض التوزيعات [القديمة جدًّا]. المزيد من المعلومات في http://www.solucorp.qc.ca/linuxconf/.

- Webmin: هذه أداة أخرى (webmin-core, webmin-dhcp, webmin-inetd, webmin-sshd packages، إلخ) تجعل من الممكن ضبط وإضافة نواحٍ متعلقة بالشبكة عبر واجهة وِب (يجب أن نكون قد ثبتنا الخادم أباتشي على سبيل المثال). رغم أنها ما تزال تطوّر في العديد من التوزيعات، فهي غير مضمّنة مبدئيًّا. لمزيد من المعلومات، أرجو زيارة http://www.webmin.com. لتشغيل الأداة بعد تثبيتها، من المتصفح اطلب العنوان https://localhost:10000، والتي ستطلب منك قبول شهادة SSL واسم المستخدم (المستخدم الجذر في البداية) وكلمة المرور المرتبطة به.

- system-config-*: في فيدورا، هناك العديد من الأدوات الرسومية التي تدعى -system-config متبوعة بكلمة ما، حيث تشير تلك الكلمة إلى ما أعدت له هذه الأداة. وبشكل عام، إذا كنا في بيئة رسومية، فيمكننا الوصول إلى كل منها عبر قائمة؛ لكن كلًّا من هذه الأدوات يعني أن لدينا عنصراً في القائمة. هناك أداة مركزية تجمع كل أدوات -system-config، وهي system-config-control في مدخلة واحدة في القائمة، وواجهة رسومية واحدة يمكننا أن نختار منها عبر مجموعة من الأيقونات. من أجل هذا، علينا أن نذهب إلى "تطبيقات" ثم إلى "إضافة وإزالة البرمجيات"[10] وستعمل بالواجهة الرسومية – في وضع المستخدم الجذر – واجهة برمجية إدارة الحزم[11] (يجب أن تكون مستودعات Fedora-Extras

---

10 الواجهة الحالية في فيدورا هي جنوم 3، مما يعني عدم وجود قوائم في الوضع المبدئي. يمكن استخدام منطقة الأنشطة للبحث في البرمجيات المتاحة، أو التحويل إلى النمط التقليدي (الكلاسيكي) من إعدادات الواجهة. للمزيد حول هذه النقطة، يمكن البحث في المواقع المتعلقة بلينكس وواجهاته، مثل مجتمع لينكس العربي والمواقع المشابهة.

11 الكتّاب الأصل يذكر برمجية Pirut الأثرية. الأدوات البديلة عنها هي yumex و Gnome PackageKit و KpackageKit، إضافة إلى البرمجية الأساسية التي تعمل من سطر الأوامر وهي yum.

مفعّلة). يمكن البحث في البرمجيات المتاحة في برمجية إدارة الحزم باستخدام – على سبيل المثال – *--system-config، اختر system-config-control ثم اضغط زر "تطبيق". ومثلها مثل الخيارات الأخرى، يمكننا أن نضبط تقريباً كل مزايا الشبكة والخدمات هنا.

- NetworkManager: إنها أداة تجعل من الممكن إدارة الشبكات اللاسلكية والسلكية بسهولة وبساطة دون أيّ تعقيدات، ولكنها ليست الأفضل للخوادم (فقط للأجهزة المكتبية). تثبيت الأداة سهل جدًّا: apt-get install network-manager-xx حيث xx يمكن أن تكون جنوم أو كدي اعتماداً على الواجهة المثبتة. لضبط الأداة، علينا أن نعيئ كل المدخلات (في دبيان) etc/network/interfaces/ عدا واجهة الإرجاع – على سبيل المثال-- عبر ترك مايلي فقط:

```
auto lo
iface lo inet loopback
```

هذه الخطوة ليست إجبارية، ولكنها تجعل عملية اكتشاف الشبكات/الواجهات أسرع. في دبيان، هناك خطوة إضافية يجب اتخاذها، حيث يجب على المستخدم أن يكون مضمناً في المجموعة netdev لأسباب تتعلق بالصلاحيات. لعمل ذلك، علينا تنفيذ (كالمستخدم الجذر، وإن لم يكن فبإضافة sudo إلى بداية السطر) adduser current_user netdev وأعد تشغيل النظام (أو أعد تشغيل الشبكة بتنفيذ etc/init.d/networking restart/ والخروج ثم الولوج مجدّداً، بحيث يصير المستخدم الحالي مضمناً في المجموعة netdev).

- أدوات أخرى: (بعضها مشروح في الجزء المتعلق بالأمن) Nmap (للتفحص والمراقبة لأغراض تتعلق بأمن الشبكة)، و Nessus (تقييم أمن الشبكة عن بعد)، و Wireshark [الموقع الرسمي] http://www.wireshark.org/download.html (سابقاً: Ethereal) (وهو محلل موافيق الشبكة)، و snort (نظام اكتشاف الاختراق intrusion detection system – IDS)، و Netcat (أداة بسيطة وقوية لتصحيح واكتشاف الشبكة، و TCPDump (مراقبة الشبكات والحصول على المعلومات)، و Hping2 (تنشئ وترسل حزم ICMP/TCP/UDP لتحليل كيفية عمل الشبكة).

الأنشطة

١) عرف أوضاع الشبكة التالية:

أ- جهاز معزول.

ب- شبكة محلية صغيرة (4 أجهزة، عبارة واحدة).

ت- شبكة محلية مقسومة إلى قسمين (مجموعتان كل منهما بها جهازان، مع موجّه لكل مجموعة، وعبارة رئيسية).

ث- شبكتان محليتان متصلتان ببعضهما (مجموعتان تتكون كل منهما من جهازين + عبارة لكل من الشبكتين).

ج- جهازان متصلان عبر شبكة افتراضية خاصة. اذكر مزيا وعيوب كل إعداد، ولأي نوع من المعماريات هي مناسبة، وما المعاملات المهمة التي نحتاجها.

٢) اضبط الشبكة في الخيارات "أ" و "ب" و "ث".

ملحق: التحكم بالخدمات المرتبطة بشبكة في فيدورا

من النواحي المهمة لكل الخدمات هي كيفية بدئها. تتضمن فيدورا مجموعة من الأدوات لإدارة مراقبات الخدمات (بما فيها المتعلقة بالشبكة). كما رأينا في الجزء المتعلق بالإدارة المحلية، فإن مستوى التشغيل هو وضع التشغيل الذي سيحدّد أي المراقبات سيتم تشغيلها. يمكننا في فيدورا أن نجد: مستوى التشغيل الأول runlevel 1 (طور المستخدم الوحيد)، والثاني (متعدد المستخدمين)، والثالث (متعدد المستخدمين مع شبكة)، والخامس (الواجهة الرسومية إضافة لما في المستوى الثالث). بالعادة، نشغّل المستوى 5، أو 3 إذا لم نكن نريد أيّ واجهات رسومية. من أجل تحديد المستوى الذي يتم تنفيذه، يمكننا أن نستخدم sbin/runlevel/، ولمعرفة مستوى التشغيل الذي سيعمل مبدئيًّا cat /etc/inittab | grep :initdefault والتي ستعطينا معلومات مثل id:5:initdefault (يمكننا أيضاً ان نحرر etc/inittab/ لتغيير القيمة المبدئية).

لرؤية الخدمات التي تعمل، يمكننا أن نستخدم sbin/chkconfig --list/، ولإدارتها يمكننا استخدام system-config-services في الوضع الرسومي، أو ntsysv في سطر الأوامر. لتفعيل خدمات منفردة، يمكننا أن نستخدم chkconfig، على سبيل المثال، الأمر التالي يفعل خدمة crond للمستويات 3 و 5: sbin/chkconfig --level 35 crond on/. بغض النظر عن الطريقة التي بدأت بها الخدمات، يمكننا أن نستخدم sbin/service --status-all/ أو بشكل منفرد sbin/service/crond status لرؤية حالة كل خدمة. ويمكنا أيضاً إدارة ذلك (start, stop, status, reload, restart)، على سبيل المثال service crond stop لإيقافها، أو service crond restart لإعادة تشغيلها.

من الضروري عدم تعطيل الخدمات التالية (ما لم تكن تعي ما تفعله): **acpid, haldaemon, messagebus, klogd, network, syslogd**. أهم الخدمات المرتبطة بالشبكة (رغم أن هذه ليست قائمة شاملة، وأن بعض الخدمات لم تذكر، إلا أن معظم الخدمات مذكورة هنا) هي:

NetworkManager, NetworkManagerDispatcher: هو مراقب يمكننا بواسطته تغيير الشبكات بسهولة (Wifi وإثرنت بشكل أساسي). إذا كان لدينا شبكة واحدة فقط، فليس من الضروري أن يتم تشغيلها.

avahi-daemon, avahi-dnsconfd: من تطبيقات zeroconf وهو مفيد لاكتشاف الأجهزة والخدمات على

الشبكات المحلية دون DNS (وهو مماثل لـ mDNS).

bluetooth, hcid, hidd, sdpd, dund, pand: شبكة بلوتوث لاسلكية للأجهزة المحمولة (ليست Wifi 802,11). على سبيل المثال، لوحات المفاتيح، والفأرة، والهواتف، والسماعات، وسماعات الرأس، إلخ.

capi, isdn: شبكة تعتمد على عتاد ISDN.

iptables: خدمة جدار الحماية المعيارية في لينكس. وهي ضرورية للأمن إذا كان لدينا اتصال بالشبكة (شبكة سلكية، أو DSL، أو T1 [مثلاً]).

ip6tables: كما في سابقتها، لكن للموافيق والشبكات المعتمدة على IPv6.

netplugd: يمكن لـ netplugd أن يراقب الشبكة وينفذ أوامر عندما تتغير الحالة.

netfs: تستخدم لضم أنظمة الملفات تلقائيًا عبر الشبكة (NFS, Samba، إلخ) أثناء الإقلاع.

nfs, nfslock: هذه مراقبات معيارية لمشاركة أنظمة الملفات عبر الشبكة في أنظمة التشغيل للأنوع يونكس/لينكس/BSD.

ntpd: خادم الوقت والتاريخ عبر الشبكة.

portmap: خدمة تكميلية لـ NFS (مشاركة الملفات) و/أو NIS (الاستيثاق).

rpcgssd, rpcidmapd, rpcsvcgssd: مستخدمة في NFS v4 (الإصدار الجديد من NFS).

sendmail: يمكن أن تستخدم هذه الخدمة لإدارة البريد (MTA)، أو لدعم خدمات مثل IMAP أو POP3.

smb: هذا الخادم يجعل مشاركة الملفات مع أنظمة وندوز ممكنًا.

sshd: تسمح sshd للمستخدمين الآخرين بالاتصال تفاعلياً وبشكل آمن مع الجهاز المحلي.

yum-updatesd: خدمة التحديث عبر الشبكة لفيدورا.

xinetd: خدمة بديلة لـ inetd تقدم مجموعة من المزايا والتحسينات، كتشغيل عدة خدمات عبر نفس المنفذ على سبيل المثال (هذه الخدمة قد لا تكون مثبتة مبدئيًّا).

إدارة الخوادم

د. رِمو سُـپي بُلدريتو

مقدمة

إن الاتصال المتبادل بين الأجهزة، والسرعة العالية للاتصالات يعني أنه الممكن أن تكون الموارد المستخدمة في أماكن مختلفة عن المكان الجغرافي للمستخدم. يونكس (وجنو/لينكس بالطبع) هو بالأحرى أفضل مثال على هذه الفلسفة، لأنه ومنذ بداياته كان التركيز فيه دائماً على مشاركة الموارد واستقلالية الأجهزة. لقد تم تحقيق هذه الفلسفة بإنشاء شيء صار الآن شائعاً جدًّا: الخدمات. الخدمة مورد (قد تكون أو لا تكون عامّة)، مما يجعل من الممكن الحصول على معلومات، أو مشاركة بيانات، أو ببساطة معالجة البيانات عن بعد تحت ظروف معينة. هدفنا هو تحليل الخدمات التي تجعله مناسباً للشبكة. عامةً يكون في الشبكة جهاز (أو عدة أجهزة، اعتماداً على الإعداد) تجعل من الممكن تبادل المعلومات مع كل العناصر الأخرى. تدعى هذه الأجهزة خوادم، وتحوي مجموعة برامج تجمع المعلومات بشكل مركزي وتجعل الوصول إليها سهلاً. تفيد هذه الخدمات في تقليل التكاليف وزيادة إتاحة المعلومات، لكن علينا أن نتذكر أن الخدمات المركزية تتضمن بعض العيوب أيضاً، حيث أنها يمكن أن ينقطع اتصالها فتدع المستخدمين دون خدمة.

يفترض أن تصمم الخوادم بحيث يكون لها مرايا لحلّ هذا الإشكال.

يمكن تصنيف الخدمات إلى مجموعتين: الخدمات التي تربط الحواسيب بالحواسيب، وتلك التي تربط المستخدمين بالحواسيب. في الحالة الأولى، الخدمات هي تلك التي تحتاجها حواسيب أخرى، بينما في الثانية، الخدمات هي تلك التي يحتاجها المستخدمون (رغم هذا فهناك خدمات يمكن أن تندرج تحت كلي التصنيفين). في المجموعة الأولى، هناك خدمات الأسماء، مثال نظام أسماء النطاقات DNS،و خدمة معلومات المستخدمين NISYP، دليل معلومات LDAPK، أو الخدمات للتخزين في الخوادم الوسيطة. في المجموعة الثانية، لدينا خدمات الاتصال التفاعلي والتنفيذ عن بعد (SSH, Telnet)، ونقل الملفات FTP، وتبادل المنعلومات على مستوى المستخدم مثل البريد الإلكتروني (MTA, IMAP, POP)، والأخبار، والشبكة العنكبوتية، والموسوعات (wiki)، والملفات NFS. لعرض كل إمكانات جنو/لينكس – دبيان و فيدورا – سنعرض كلاً من هذه الخدمات بالحد الأدنى من الإعدادات التي تعمل، ولكن دون تجاهل النواحي المتعلقة بالأمن والاستقرار.

1 أنظمة أسماء النطاقات DNS

إن وظيفة خدمة DNS (كما شرحنا في الوحدة المتعلقة بإدارة الشبكة) هي ترجمة أسماء الأجهزة (مفهومة ويسهل على المستخدمين تذكرها) إلى عناوين IP والعكس.

مثال

عندما نسأل عن عنوان إنترنت لـ pirulo.remix.com، سيستجيب الخادم بـ 192.168.0.1 (تعرف العملية بـ mapping)، وبشكل مشابه، عندما نطلب عنوان IP، ستستجيب الخدمة باسم الجهاز (يعرف بـ reverse mapping).

> إن نظام أسماء النطاقات DNS معمارية شجرية تجنب تكرار المعلومات وتجعل البحث أسهل. ولهذا السبب، فإن وجود DNS وحيد ليس له معنى، مالم يكن جزءاً من المعمارية.

يعرف التطبيق الذي يقدم هذه الخدمة بالاسم named، وهو مضمّن في معظم توزيعات جنو/لينكس (usr/sbin/named/) وهو جزء من الحزمة التي تدعى bind (حالياً الإصدار 9[1]) والذي يديره اتحاد برمجيات الإنترنت Internet Software Consortium – ISC. الـ DNS ببساطة قاعدة بيانات، مما يعني أن الناس الذين يعدلونها يجب أن يعرفوا معماريتها، عدا هذا ستتأثر الخدمة. احتياطاً، يجب أخذ احتياطات خاصة بحفظ نسخ عن الملفات لتجنب أي انقطاع للخدمة. الحزمة في دبيان تأتي بالاسم bind و bind-doc. الإعدادات مشابهة لما في فيدورا، ولكن سيكون علينا تثبيت bind و bind-utils و caching-nameserver وهذا يمكن عمله عبر yum على سبيل المثال.

1.1 خادم حفظ الأسماء

بداية، سنضبط خادم DNS لتلقي استعلامات، الذي سيعمل على حفظ استعلامات الأسماء (خادم تلقي وحفظ فقط)، وبعبارة أخرى، في المرة الأولى، سيتم استشارة الخادم المناسب، و ذلك لأننا نبدأ بقاعدة بيانات لا تحوي معلومات، لكن في كل المرات اللاحقة، سيستجيب خادم حفظ الأسماء، مع تضاؤل في وقت الاستجابة. لضبط خادم حفظ الأسماء، نحتاج الملف etc/bind/named.conf/ (في دبيان)، الذي فيه ما يلي (التعليقات الأساية في الملف – التي تبدأ بـ// – تم مراعاتها):

---

1 الإصدار العاشر ما زال قيد التطوير.

```
options {
directory "/var/cache/bind";
        // query-source address * port 53;
        // forwarders {
        //      0.0.0.0;
        // };
        auth-nxdomain no; # conform to RFC1035
        };
// prime the server with knowledge of the root servers}
zone "." {
        type hint;
        file "/etc/bind/db.root"; };
        // be authoritative for the localhost forward and reverse zones, and for
        // broadcast zones as per RFC 1912
        }
zone "localhost" {
        type master;
        file "/etc/bind/db.local";
        };

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
        };

        zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
        };

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
        };
// add entries for other zones below here
}
```

تشير جملة directory إلى المكان الذي سنجد فيه بقية ملفات الإعداد (وهو في حالتنا /var/cache/bind/). الملف /etc/bind/db.rootسيحوي شيئاً مشابهاً لما يلي (تظهر فقط السطور الأولى وهي سطور التعليق التي تبدأ بالفاصلة المنقوطة

";"، ويجب الحذر فيما يتعلق بالنقط "."") في بداية بعض الملفات – يمكن الحصول عليها وتحديثها مباشرة من الإنترنت – :

```
...
; formerly NS.INTERNIC.NET
;
. 3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 A 198.41.0.4
;
; formerly NS1.ISI.EDU
;
. 3600000 NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 A 128.9.0.107
;
; formerly C.PSI.NET
;
. 3600000 NS C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000 A 192.33.4.12
;
...
```

يعرض هذا الملف خوادم الأسماء الرئيسية (root name servers) في العالم. هذه الخوادم تتغير، مما يعني أنه يجب تحديث الملف من الإنترنت باستمرار. الأقسام التالية هي المناطق، المنطقتان localhost و 127.in-addr.arpa – التي تربط الملفات بالمجلدين /etc/bind/db.local/ و /etc/bind/db.127/ تشير إلى الحلّ المباشر والعكسي للواجهة المحلية. المناطق التالية هي لمناطق البثّ (انظر إلى RFC1912) والمناطق التي يفترض أن تضاف في النهاية. على سبيل المثال، الملف db.local يمكن أن يكون (الفاصلة المنقوطة ";" تعني تعليقاً):

```
; BIND reverse data file for local loopback interface
$TTL 604800
@       IN      SOA     ns.remix.bogus.     root.remix.bogus. (
                1                           ; Serial
                604800                      ; Refresh
                86400                       ; Retry
                2419200                     ; Expire
                604800)                     ; Negative Cache TTL
@               IN      NS                  ns.remix.bogus.
1.0.0   IN      PTR     localhost.
```

سنشرح كيف تستخدم لاحقاً. الخطوة التالية هي وضع خادم الأسماء في etc/resolv.conf/:

```
search subdomain.your-domain.domain your-domain.domain
# for example search remix.bogus bogus
nameserver 127.0.0.1
```

حيث سيكون علينا أن نغير subdomain.your-domain.domain إلى القيم المناسبة. يوضح سطر البحث أي النطاقات سيتم البحث فيها عن أي مضيف يرغب بالاتصال (يمكن تغيير search إلى domain، لكنهما يتصرفان بطريقة مختلفة) ويحدد خادم الأسماء عنوان خادم الأسماء (وهي في هذه الحالة جهازك نفسه، والذي ستتم فيه عملية التسمية). يتصرف البحث كما يلي: إذا كان العميل يبحث عن جهاز اسمه pirulo، فسيتم البحث أولاً في pirulo.subdomain.your-domain.domain، ثم pirulo.your-domain.domain،وفي النهاية pirulo. هذا يعني بأن البحث سيأخذ بعض الوقت؛ ولكن إذا كان pirulo في subdomain.your-domain.domain، فليس من الضروري دخول البقية.

الخطوة التالية هي بدء named والنظر في نتائج التنفيذ. لتشغيل المراقب، يمكننا أن نستخدم النص البرمجي للتشغيل etc/init.d/bind9 start/ مباشرة (إذا كان named يعمل بالفعل، اذهب إلى etc/init.d/bind9 reload/) أو – إن لم يكن – usr/sbin/named/. إذا نظرنا إلى سجلات النظام في var/log/daemon.log/، فسنرى شيئاً مشابهاً لما يلي:

```
Sep 1 20:42:28 remolix named[165]: starting BIND 9.2.1 \\
Sep 1 20:42:28 remolix named[165]: using 1 CPU \\
Sep 1 20:42:28 remolix named[167]: loading configuration from
'/etc/bind/named.conf'
```

بدء تشغيل النظام ورسائل الخطأ ستظهر هنا (إذا كانت هناك أخطاء، وفي هذه الحالة يجب تصحيحها وتشغيل الخدمة

مرة أخرى) يمكننا الآن التحقق من الإعداد بأوامر مثل nslookup (الأصيل والسهل، ولكنه أثريّ كما يقول مبرمجوه)، أو host، أو dig (مستحسن). مخرجات dig -x 127.0.0.1 ستكون مشابهة لما يلي:

```
# dig -x 127.0.0.1
;; <<>> DiG 9.2.1 <<>> -x 127.0.0.1
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31245
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION: ;1.0.0.127.in-addr.arpa. IN PTR
;;ANSWER SECTION: 1.0.0.127.in-addr.arpa. 604800 IN PTR localhost.
;; AUTHORITY SECTION: 127.in-addr.arpa. 604800 IN NS ns.remix.bogus.
;; Query time: 1 msec
;; SERVER: 127.0.0.1 #53(127.0.0.1)
;; WHEN: Mon Sep 1 22:23:35 2003
;; MSG SIZE rcvd: 91
```

والتي يمكننا أن نرى فيها أن الاستعلام استغرق 1 ميلي ثانية. إذا كان لديك اتصال بالإنترنت، يمكنك البحث عن جهاز ضمن نطاقك وأن ترى أداء الخادم وسلوكه. في BIND9 هناك lwresd (اختصاراً لعبارة lightweight resolver daemon)، وهو المراقب الذي يقدم خدمات التسمية للعملاء الذين يستخدمون مكتبة BIND9 lightweight resolver. هو بالأساس خادم حفظ مؤقت caching (مثل الذي ضبطناه) يقوم بالاستعلام باستخدام BIND9 lightweight resolver protocol بدلاً من ميفاق DNS. يستمع هذا الخادم عبر الواجهة 127.0.0.1 (مما يعني أنه يستمع للعمليات في المضيف المحلي) في UDP والمنفذ 921. يتم إلغاء تعمية طلبات العملاء وحلها باستخدام ميفاق DNS. عند الحصول على استجابات، يرمّزها lwresd بهيئة lightweight ويعيدها إلى العميل الذي طلبها.

في النهاية، كما ذكرنا، تستخدم النواة عدة مصادر معلومات، يتم الحصول على المتعلقة بالشبكة منها من etc/nsswitch.conf/. يشير هذا الملف إلى المكان الذي نحصل منه على مصدر المعلومات وهناك قسم لأسماء الأجهزة وعناوين IP لها، مثل:

```
hosts: files dns
```

يشير هذا السطر (إن لم يكن موجوداً، فيفترض أن تتم إضافته) إلى أنه من كان يحتاج – أيًّا يكن – اسم جهاز أو

عنوان إنترنت عليه أولاً أن يتفحص etc/hosts/، ومن ثم في DNS اعتماداً على النطاقات المذكورة في etc/resolv.conf/.

1.2 Forwarders

في الشبكات ذات الأحمال العالية، من الممكن موازنة زحام البيانات باستخدام القسم المتعلق بـ Forwarders. إذا كان لدى مزود خدمة الإنترنت ISPخادم أسماء مستقر واحد أو أكثر، فمن المحبذ استخدامها لتخفيف الطلبات على الخادم. من أجل هذا، علينا حذف التعليق (//) من كل سطر في قسم الـ forwarders من الملف etc/bind/named.conf، وتغيير 0.0.0.0 إلى عناوين خوادم الأسماء لمزود الخدمة. هذا الإعداد ينصح به عندما يكون الاتصال بطيئاً، عند استخدام مودم على سبيل المثال.

1.3 إعداد نطاق خاص

يعالج DNS معمارية شجرية، ويعرف الأصل فيها بـ"." (انظر إلى etc/bind/db.root/). تحت الـ"." هناك TLDs (أي نطاقات المستوى الأعلى top level domains)، مثل org, com, edu, net، إلخ. عند البحث في خادم، إذا لم يعرف الخادم الإجابة، سيتم البحث في الشجرة بفروعها إلى أن يتم إيجاده. كل نقطة في العنوان (على سبيل المثال، pirulo.remix.com) تشير إلى فرع مختلف في شجرة DNS ومجال مختلف للطلب (أو الاستجابة 1) التي سيتم اتباعها بفروعها من اليسار إلى اليمين.

وهناك ناحية أخرى هامة – بعيداً عن النطاق – وهي in-addr.arpa (أي reverse mapping)، والتي يتم تضمينها أيضاً كالنطاقات، وتخدم في الحصول على أسماء عند الطلب بعنوان IP. في هذه الحالة، تكتب الأسماء بطريقة معكوسة، اعتماداً على النطاق. إذا كان pirulo.remix.com هو 192.168.0.1، فستكون مكتوبة 1.0.168.192 – اعتماداً على pirulo.remix.com. علينا عند إذٍ ضبط النطاق الأصل remix.bogus في الملف etc/bind/db.127/:

```
; BIND reverse data file for local loopback interface
$TTL 604800
@	IN	SOA	ns.remix.bogus.	root.remix.bogus. (
		1			; Serial
		604800			; Refresh
		86400			; Retry
		2419200			; Expire
		604800)			; Negative Cache TTL
@		IN	NS		ns.remix.bogus.
1.0.0	IN	PTR	localhost.
```

يجب أخذ النقطة "." بعين الاعتبار في نهاية أسماء النطاقات. إن أصل هيكلية منطقة يحدده تعريف المنطقة، وهي في حالتنا in-addr.arpa.127. هذا الملف (db.127) يحوي 3 سجلات: SOA, NS, PTR. الأول SOA (اختصاراً لعبارة start of authority) يجب أن يكون موجوداً في كل ملفات المناطق، في البداية - بعد TTL و @ التي تحدد أصل النطاق؛ NS هو خادم الأسماء name server للنطاق، و PTR وهو مؤشر أسماء النطاقات (domain name pointer) وهو المضيف 1 في الشبكة الفرعية 127.0.0.1 ويسمى المضيف المحلي local host. هذا ملف السلسلة 1، و root@remix.bogus (آخر فراغ في سطر SOA) هو المسؤول عنه. يمكننا الآن أن نشغل named كما هو مذكور أعلاه، ويمكننا - باستخدام dig -x 127.0.0.1 - أن نرى كيف يعمل (مماثلاً لما عرضناه سابقاً).

سيكون علينا بعد ذلك أن نضيف منطقة أخرى في named.conf:

```
zone “remix.bogus” {
        type master;
        notify no;
        file “/etc/bind/remix.bogus”;
        };
```

يجب علينا أن نتذكر أنه في named يظهر النطاق دون نقطة في نهايته. سنضع في الملف remix.bogus المضيفين الذين سنكون مسؤولين عنهم:

```
; Zone file for remix.bogus
$TTL 604800
@       IN      SOA             ns.remix.bogus.       root.remix.bogus. (
        199802151               ; serial, todays date + todays serial
        604800                  ; Refresh
        86400                   ; Retry
        2419200                 ; Expire
        604800 )                ; Negative Cache TTL
@       NS      ns              ; Inet Address of name server
        MX      10              mail.remix.bogus.     ; Primary Mail Exchanger
localhost       A               127.0.0.1
ns              A               192.168.1.2
mail            A               192.168.1.4
                TXT "Mail Server"
ftp             A               192.168.1.5
                MX              10 mail
www             CNAME           ftp
```

يظهر هنا سجل MX – سجل تبادل البريد Mail eXchanger – جديد. إنه المكان الذي يرسل إليه البريد الذي يصل – someone@remix.bogus – وسيتم إرسالها إلى mail.remix.bogus (يشير الرقم إلى الأولوية إذا كان لدينا أكثر من MX). تذكر دوماً النقطة الضرورية في ملفات المناطق في نهاية النطاق (إذا لم تكن مدخلة، سيضيف النظام نطاق SOA إلى النهاية، والذي قد ينقل mail.remix.bogus – على سبيل المثال – إلى mail.remix.bogus.remix.bogue والذي لن يكون صحيحاً). يستخدم CNAME (أو canonical name) لإعطاء جهاز ما اسماً مستعاراً واحداً أو أكثر. كما هو لدينا الآن، سنكون قادرين (بعد تنفيذ /etc/init.d/bind9 reload على اختبار – على سبيل المثال – dig.remix.bogus.

الخطوة الأخيرة هي ضبط منطقة المعكوسات inverse، وبعبارة أخرى، من أجل أن يكون من الممكن تحويل عناوين IP إلى أسماء بإضافة مناطق جديدة مثلاً:

```
zone "192.168.1.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/192.168.1";
};
```

والملف /etc/bind/192.168.1 المشابه لسابقه:

```
$TTL 604800
@       IN      SOA             ns.remix.bogus.       root.remix.bogus. (
        199802151               ; serial, todays date + todays serial
        604800                  ; Refresh
        86400                   ; Retry
        2419200                 ; Expire
        604800 )                ; Negative Cache TTL
@       NS                      ns.remix.bogus.
2       PTR                     ns.remix.bogus
4       PTR                     mail.remix.bogus
5       PTR                     ftp.remix.bogus
```

يمكن اختبار ذلك مجدداً باستخدام dig -x 192.168.1.4. علينا أن نتذكر بأن هذه الأمثلة على عناوين خاصة، وبعبارة أخرى، ليست عناوين على الإنترنت. وهناك نقطة مهمة أخرى، وهي أنه يجب علينا أن لا ننسى الإشعار، وإلا فستنتشر اختباراتنا على الـDNS إلى الخوادم عبر شجرة DNS (ويمكن حتى أن تغير DNS المزود أو المؤسسة لدينا). يجب أن يتم تغييرها فقط عندما نكون واثقين من أنها تعمل ومتأكدين من التغيرات التي نريد عملها. للنظر إلى مثال حقيقي، فضلاً انظر إلى DNS-HOWTO في http://tldp.org/HOWTO/DNS-HOWTO-7.html.

بمجرد أن ننشئ خادماً رئيسيًّا master server، يجب علينا إنشاء خادم تابع slave من أجل الأمن، وأن يكون مطابقاً للرئيسي، باستثناء أن المنطقة في مكان النوع كلمة slave (أي "تابع") بدلاً من master، إضافة إلى عنوان الخادم الرئيسي. على سبيل المثال:

```
zone "remix.bogus" {
        type slave;
        notify no;
        masters [192،168،1،2];
};
```

NIS (YP) 2

من أجل تسهيل الإدارة وجعل النظام أكثر سلاسة للمستخدم، فهناك في الشبكات المختلفة الأحجام التي تشغّل جنو/لينكس (أو Sun Solaris أو أي نظام تشغيل آخر يدعم هذه الخدمة) خدمات معلومات الشبكة Network Information Services أو NIS (أو الاسم الأصلي الذي ابتكرته Sun وهو Yellow Pages أو YP). يمكن لجنو/لينكس أن يقدم دعماً كخادم وعميل NIS ويمكن أن يكون عميلاً لـ+NIS، وهو إصدار آمَن وأكثر تخصيصاً من NIS. المعلومات التي يمكن توزيعها في NIS هي: المستخدمون (أسماء الولوج)، وكلمات المرور (/etc/passwd)، ومجلدات المستخدمين (مجلد المنزل)، ومعلومات المجموعات (/etc/group) والتي بها ميزة أنه يمكن للمستخدم الاتصال - من أي جهاز عميل أو من الخادم نفسه - بنفس الحساب وكلمة المرور وإلى نفس المجلد (ولكن يجب أن يكون المجلد قد تم ضمّه مسبقاً على كل الأجهزة باستخدام NFS أو باستخدام خدمة automount).

> معمارية NIS من النوع عميل—خادم، وبعبارة أخرى هناك خادم ستكون عليه كل قواعد البيانات وبعض العملاء الذين سيستخدمون هذه البيانات وستبدو للمستخدم كما لو كانت محلية. ولهذا السبب، يجب أن نأخذ بعين الاعتبار ضبط خوادم "التعزيزيات" (وتسمى الخوادم الثانوية) بحيث لا يُحجب المستخدمون عندما لا يكون الخادم الرئيسي متاحاً. تدعى هذه المعمارية بمعمارية الخوادم المتعددة multiple server architecture (خادم رئيسي+مرايا—عملاء).

2.1 كيف نبتدئ عميل NIS محلي في دبيان؟

وجود حاسوب محلي يعني إضافة الحاسوب إلى نطاق NIS موجود:

- أولاً، يجب علينا التحقق من أن حزم netbase (شبكة TCP/IP أساسية) و portmap (وهي خدمة تقوم بتحويل أرقام RPC إلى منافذ DARPA الضرورية للبرامج التي تشغّل RPC، بما فيها NFS و NIS)، و nis (معينة) قد تم تثبيتها. ننصح باستخدام الأمر kpackage، أو مباشرة عبر apt-get (يمكننا التحقق مما إذا كان موجوداً أو لا بالأمر

apt-cache pkgnames) في الوضع النصيّ. عند تثبيت حزمة NIS، ستُسأل عن نطاق (اسم نطاق NIS). إنه اسم يصف مجموعة الأجهزة التي ستستخدم NIS (ليس اسم مضيف). أبق في بالك أن اسم النطاق NISPirulo يختلف عن Nispirulo. يمكنك أن تستخدم الأمر nisdomainname لضبط هذا النطاق، وهو نطاق سيتم حفظه في /proc/sys/kernel/domainname.

- بداية، علينا تشغيل الخدمة portmap بالأمر:

```
/etc/init.d/portmap start
```

يمكننا التحقق مما إذا كانت تعمل rpcinfo -p.

- خادم NIS هذا ليس محلياً، علينا أن نستخدم الأمر ypbind. يستخدم الأمر ypbind لإيجاد خادم نطاق محدد، سواء عبر البثّ broadcast (لا ينصح به، لأنه غير آمن)، أو عبر البحث عن الخادم المشار إليه في ملف الإعداد /etc/yp.conf (مستحسن). للملف /etc/yp.conf الصيغة التالية:

  - domain nisdomain server hostname: يشير إلى أنه يفترض أن يُستخدَم hostname[2] لـ nisdomain. يمكن أن تكون لدينا أكثر من مدخلة من هذا النوع لنفس النطاق.

  - domain nisdomain broadcast: يشير إلى أنه يفترض أن يُستخدم البثّ في الشبكة المحلية لاكتشاف خادم ذي نطاق NIS المشار إليه.

  - ypserver hostname: يشير إلى أنه يفترض أن يُستخدم hostname كخادم. يُنصح باستخدام هذا السطر (أي الأمر ypserver) الذي يفرض علينا استخدام عنوان IP لخادم NIS. إذا تم تحديد الاسم، فتأكد من أنه يمكن العثور على العنوان عبر DNS أو أنه يظهر في الملف /etc/hosts، وإلا فسيتم حجب العميل.

- شغّل الخدمة بتنفيذ:

---

2 استخدم الكاتب هنا hostname كمثال على اسم المضيف (وهذا ينطبق على الأمثلة التالية)، كما استخدم nisdomain كمثال على اسم نطاق nis.

```
/etc/init.d/nis stop
```

ثم:

```
/etc/init.d/nis start
```

- يفترض أن تعمل هذه الأوامر. سيعمل عميل NIS (يمكن التأكد من ذلك بالأمر rpcinfo أو localhost ypbind التي ستظهر إصدارين من الميفاق الذي يعمل) أو يمكننا أن نستخدم الأمر ypcat mapname (على سبيل المثال ypcat passwd الذي سيظهر مستخدمي NIS المعرفين في هذا الخادم) حيث علاقة الـ mapnames بالجداول في قاعدة بيانات NIS معرفة في /var/yp/nicknames.

2.2 ما الموارد التي يجب تحديدها لاستخدام NIS ؟

لنفترض أننا ثبتنا إحدى أحدث توزيعات دبيان، والتي تدعم إصداراً حديثاً من libc (كما في الإصدارات الحديثة من فيدورا) ونرغب بضبط النظام بحيث يتمكن مستخدم جهاز العميل من الوصول إلى المعلومات التي في الخادم. في هذه الحالة، علينا إرسال طلب الولوج إلى قاعدة البيانات المناسبة كالآتي:

١) التحقق من /etc/nsswitch.conf والتأكد من أن مدخلات كل من passwd, group, shadow, netgroup مشابهة لما يلي:

```
passwd: compat
group: compat
shadow: compat
netgroup: nis
```

انظر إلى man nsswitch.conf لمعرفة صياغة هذا الملف.

٢) والسطر التالي في جهاز عميل NIS في نهاية الملف /etc/passwd (سيعني هذا أنه إذا لم يكن العميل محلياً، فسيتم سؤال خادم NIS):

(إشارة جمع واحدة و 6 أزواج من النقاط العمودية)   +::::::

٣) علينا أن نتذكر بأنه يمكننا استخدام إشارة الجمع "+" وعلامة الاستفهام "?" في الملف etc/passwd/ قبل كل اسم مستخدم لتضمين أو تخطي ولوج هؤلاء المستخدمين. إذا كنا نستخدم كلمات مرور مع shadows (وهي طريقة آمَن، حيث لن تسمح للمستخدمين العاديين برؤية كلمات المرور المعمّاة للمستخدمين الآخرين)، فيجب إضافة السطر التالي إلى نهاية الملف etc/shadow/:

(إشارة جمع وثمانية أزواج من النقاط العمودية)   +::::::::

٤) يجب أيضاً إضافة السطر التالي إلى نهاية الملف etc/group/:

(إشارة جمع وثلاثة أزواج من النقاط العمودية)   +:::

٥) سيتم تنفيذ البحث عن مضيفين (hosts lookup) عبر DNS (وليس NIS) مما يعني أنه سيكون علينا تغيير مدخلة المضيفين في السطر التالي: hosts: files dns في الملف etc/nsswitch.conf/ – لتطبيقات Libc6. أو إذا كنا نفضل عمل هذا باستخدام NIS فسنعدل hosts: files nis. لتطبيقات libc5، علينا أن نعدل الملف host.conf بإدخال order hosts, DNS أو hosts, NIS حسب الحاجة.

في هذا الإعداد، سيكون من الممكن إنشاء اتصال محلي (عبر عميل NIS) بمستخدم غير معرف في الملف etc/passwd/، وبعبارة أخرى، بمستخدم معرف في جهاز آخر (في الخادم ypserver).

على سبيل المثال، يمكن أن ننفذ ssh -l localhost، حيث يكون المستخدم معرّفاً في ypserver.

2.3 كيف يمكننا أن ننفذ خادم NIS رئيسيّ؟

لنفترض أننا ثبتنا حزمة nis و portmap (وأن portmap يعمل) وأن قاعدة بيانات NIS تم إنشاؤها (انظر إلى القسم التالي):

- يجب علينا أن نتأكد من أن etc/hosts/ يحوي كل الأجهزة التي ستشكل جزءاً من النطاق بصيغة FQDN (أي Fully Qualified Domain Name) والتي يظهر فيها كل من عنوان IP والاسم شاملاً النطاق والاسم دون النطاق لكل جهاز (على سبيل المثال، 192,168,0,1 pirulo.remix.com pirulo). هذا ضروري فقط في الخادم،

حيث لا يستخدم NIS خدمة DNS.

- إضافة إلى هذا، علينا التأكد أيضاً من أنها موجودة في الملف etc/defaultdomain/ باسم النطاق المختار. لا تستخدم نطاق DNS الخاص بك، لكي لا تتسبب بأية مخاطر أمنية، إلا أن تضبط كلًّا من الملف etc/ypserv.securenets/ – الذي يشير إلى المواقع التي سيكون من الممكن أن يتصل منها العملاء بزوج network/netmask – والملف etc/ypserv.conf/ – الذي يحوي تحكماً أكثر تفصيلاً – جيداً، وذلك لأنها تحدّد ما يمكن لكل من المضيفات أن يصل إليه من الخرائط maps، على سبيل المثال: passwd.byname o shadow.byname.

- تحقق من أن NISSERVER = master موجودة في etc/default/nis/.

- لأسباب أمنية، من الممكن إضافة رقم الشبكة المحلية إلى الملف etc/ypserv.securenets/.

- شغل الخادم بتنفيذ الأمر etc/init.d/nis stop/ ثم etc/init.d/nis start/. سيشغّل هذا الأمر الخادم ypserv ومراقب كلمة المرور yppasswdd والذي يمكن التأكد إذا كان يعمل أم لا بالأمر ypwhich -d domain.

2.4 كيف يمكننا أن نضبط خادماً ؟

يُضبط الخادم بالأمر usr/lib/yp/ypinit -m/؛ ولكن من الضروري التحقق من أن الملف etc/networks/ موجود، فهو ضروري لهذا النصّ البرمجي.

إذا لم يكن هذا الملف موجوداً، أنشئ واحداً فارغاً بالأمر touch /etc/networks. من الممكن أيضاً جعل عميل ypbind أن يُنفّذ على الخادم؛ بهذه الطريقة، كل المستخدمين الذي يدخلون عبر NIS – وذلك كما ذُكر أعلاه، بتعديل الملف etc/passwd/ – حيث سيتجاهل NIS كل المدخلات العادية قبل ::::::+ (يمكن الوصول إليها محلياً فقط)، حيث يمكن الوصول إلى البقية عبر NIS من أي عميل.

اعتبر أنه بدءاً من هذه اللحظة، الأوامر المستخدمة لتغيير كلمة المرور أو معلومات المستخدم مثل passwd, chfn,

adduser لم تعد صالحة. بدلاً من ذلك، سيكون علينا أن نستخدم أوامر مثل yppasswd, ypchsh, ypchfn. عند تغيير المستخدمين أو تعديل الملفات المذكورة أعلاه، سيكون علينا أن نعيد إنشاء جداول NIS بتنفيذ الأمر make في المجلد /var/yp لتحديث الجداول.

إن ضبط خادم تابع شبيه بضبط خادم رئيسي، باستثناء أن NISSERVER = slave في /etc/default/nis. على الخادم الرئيسي، علينا أن نشير إلى أنه يجب توزيع الجداول تلقائيًّا إلى الخوادم التابعة، وذلك بإضافة NOPUSH = false في الملف /var/yp/Makefile.

الآن علينا إخبار الخادم الرئيسي عن تابعه، وذلك بتنفيذ:

```
/usr/lib/yp/ypinit -m
```

وإضافة أسماء التوابع إلى الخادم الرئيسي. هذا سيعيد إنشاء الخرائط، لكنه لن يرسل الملفات إلى التوابع. للقيام بذلك، نفذ في التابع:

```
/etc/init.d/nis stop
/etc/init.d/nis start
```

وفي النهاية:

```
/usr/lib/yp/ypinit -s name_master_server.
```

بهذه الطريقة، سيُحمِّل التابع الجداول من الرئيسي.

يمكن أيضاً وضع ملف nis في المجلد /etc/cron.d بمحتوى مشابه لما يلي (تذكر أن تُنفِّذ chmod 755 /etc/cron.d/nis)

والتي سنتأكد بها من أن كلّ التغييرات في الخادم الرئيسي سيتم ترحيلها إلى خوادم NIS التابعة.

توصية: بعد تنفيذ adduser لإضافة مستخدم إلى الخادم، نفّذ make -C /var/yp لتحديث جداول NIS (ونفذ هذا عند حدوث أيّ تغير لخصائص مستخدم، ككلمة المرور بالأمر passwd مثلاً والتي ستغير كلمة المرور المحلية وليس كلمة مرور

NIS). للتحقق من أن النظام يعمل، وأن المستخدم مسجل في NIS، يمكنك أن تنفّذ ypmatch userid passwd حيث userid هو المستخدم المسجل مؤخراً باستخدام adduser وبعد تنفيذ make. للتأكد من أن نظام NIS يعمل، يمكن أن تستخدم النصّ البرمجي الموجود في http://tldp.org/HOWTO/NIS-HOWTO/verification.html، والتي تسمح بعمل تحقق أكثر تفصيلاً في NIS.

3 خدمات الاتصال عن بعد: telnet و ssh

3.1 telnet و telnetd[3]

إن telnet أمر (عميل) مستخدم للتواصل تفاعلياً مع مضيف آخر يشغّل المراقب telnetd. يمكن تنفيذ الامر telnet مباشرة telnet host أو تفاعليًّا عبر الأمر telnet الذي سيدخل المحثّ "telnet>"، ومن ثم – على سبيل المثال – open host. بمجرد إنشاء اتصال، يجب علينا إدخال المستخدم وكلمة المرور الذي نرغب باستخدامهما للاتصال بالنظام البعيد. هناك أوامر عديدة (في الوضع التفاعلي)، مثل open و logout و mode (يحدّد خصائص المحاكاة)، و close, encrypt, quit, set, unset أو يمكنك تنفيذ أوامر خارجية بـ"!". يمكنك أن تستخدم الملف /etc/telnetrc للإعدادات المبدئية، أو telnetrc. لإعدادات مستخدم معين (يجب أن تكون هذه الإعدادات في مجلد المنزل للمستخدم).

المراقب telnetd هو خادم ميفاق telnet للاتصال التفاعلي. يتم تشغيل telnetd عموماً باستخدام المراقب inetd ويحبّذ أن يكون tcpd wrapper (الذي يستخدم قواعد الوصول في hosts.allow و hosts.deny) مضمناً في استدعاء telnetd في الملف /etc/inetd (على سبيل المثال)، أضف سطراً يشبه:

```
telnet stream tcp nowait telnetd.telenetd /usr/sbin/tcpd /usr/bin/in.telnetd
```

لزيادة أمن النظام، فضلاً انظر إلى الوحدة المتعلقة بالأمن. في بعض التوزيعات، يمكن استبدال وظائف inetd إلى xinetd، مما يعني أنه يجب أن يكون الملف /etc/xinetd.conf مضبوطاً (انظر في الوحدة المتعلقة بإدارة الشبكات). وكذلك، إذا رغبنا بتشغيل inetd في وضع الاختبار، يمكنك استخدام الجملة /etc/init.d/inetd.real start. إذا كان الملف /etc/uissue.net موجوداً، فسيعرض telnet محتوياته عند الولوج. يمكن استخدام /etc/security/access.conf لتفعيل أو تعطيل ولوج المستخدمين، أو ولوج المضيفين، أو ولوج مجموعات المستخدمين عند اتصالهم

---

3 خدمة telnet تعدّ قديمة جدًّا، وتحوي مشاكل أمنية كثيرة، لذا لا يُنصح باستخدامها، ويُستخدم بدلاً عنها طرق اتصال أكثر أمناً، كالصدفة الآمنة SSH (القسم التالي)؛ لكن قد تضطر لاستخدامها مع أنظمة لا تدعم SSH كالأنظمة التي لا تدعم معايير POSIX مثل وندوز.

علينا أن نتذكر بأنه رغم إمكانية أن يعمل الزوج telnet-telnetd في وضع مشفّر في الإصدارات الأخيرة (نقل البيانات المشفرة، لكن يجب أن يتم تصريفها مع تفعيل الخيار ذي العلاقة)، إلا أنه أمر أثريّ (مُلغى)، وذلك بشكل أساسيّ بسبب افتقاره للأمن، لكن ما زال بالإمكان أن يُستخدم في الشبكات الآمنة أو في الأوضاع تحت السيطرة.

إذا لم يكن مثبتاً، يمكننا أن نستخدم (في دبيان) apt-get install telnetd ومن ثم التحقق من أنه تم تسجيله إما في /etc/inetd.conf أو /etc/xinetd.conf (أو في المجلدات المحددة في هذين الملفين، مثل /etc/xinetd.d/ كما أشير إليه في الملف السابق في الجملة التي تتضمن مساره). يجب أن يحوي الملف xinetd.conf أو xinetd.d/telnetd قسماً مثل (مع أي تغيير في xinetd.conf، يجب إعادة تشغيل الخدمة بالأمر service xinetd restart):

```
service telnet
{
disable = no
flags = REUSE
socket_type = stream
wait = nouser = root
server = /usr/sbin/in.telnetd
log_on_failure += USERID
}
```

بدلاً من استخدام telnetd، ننصح باستخدام SSL telnetd والتي تستبدل telnetd باستخدام تعمية واستيثاق عبر SSL، أو استخدام SSH (القسم التالي). يمكن أن يعمل SSLTelnet مع telnet بشكل عادي بكلي الاتجاهين، حيث أنه عند بدء الاتصال، يتم التحقق مما إذا كان الطرف الآخر يدعم SSL أم لا، وإن لم يكن، فسيتم الاتصال عبر ميفاق telnet العادي. المزايا مقارنة مع telnet هي أن كلمات المرور والبيانات لا تمر عبر الشبكة بوضع نصّ صَرِف، وأي شخص يستخدم – على سبيل المثال – tcpdump يمكنه رؤية محتوى المحادثة. ويمكن استخدام SSLTelnet أيضاً – مثلاً – للاتصال بخوادم وب آمنة (على سبيل المثال: https://servidor.web.org) ببساطة بتنفيذ telnet server.web.org 443.

3.2 الصدفة الآمنة Secure Shell – SSH

ينصح بالتغيير إلى استخدام SSH بدلاً من telnet و rlogin و rsh. هذه الأوامر الأخيرة غير آمنة (عدا SSLTelnet) لأسباب عديدة: أهمها هو أن كل ما يتم نقله عبر الشبكة – بما في ذلك أسماء المستخدمين وكلمات المرور – يتم نقله كنصّ صرف (ورغم أن هناك إصدارات معمّاة من telnet-telnetd، فيجب أن يتصادف أن يكون كلا الطرفين يدعمان التعمية)، وأي شخص لديه وصول إلى تلك الشبكة أو أي جزء منها سيكون قادراً على الحصول على كل المعلومات ومن ثم انتحال شخصية المستخدم. والثاني هو أن هذه المنافذ (telnet, rsh, ...) هي أول ما سيحاول المخترقون الاتصال به. يقدم ميفاق ssh (في الإصدار OpenSSH) اتصالاً معمّىً ومضغوطاً أكثر أمناً بكثير من telnet مثلاً (يُنصح باستخدام الإصدار الثاني من الميفاق). تتضمن كل التوزيعات الحالية عميل ssh وخادم sshd مبدئيًّا.

3.2.1 عميل ssh

لتنفيذ الأمر، استمر كما يلي:

ssh -l *login name host* أو ssh *user@hostname*

يمكننا عبر SSH تضمين اتصالات أخرى مثل X11 أو أي اتصال TCP/IP آخر. إذا أزلنا المعامل l-، فسيلج المستخدم إلى نفس المستخدم المحليّ وفي كلتي الحالتين سيسأل الخادم عن كلمة المرور للتحقق من هوية المستخدم. يدعم SSH أنماط استيثاق مختلفة (انظر إلى صفحات man ssh) معتمدة على خوارزميات RSA والمفاتيح العامة.

يمكن إنشاء مفاتيح التحقق من المستخدم باستخدام الأمر ssh-keygen -t rsa|dsa. ينشئ الأمر المجلد ssh. والملفين id_rsa و id_rsa.pub، وهما المفتاحان الخاص والعام على الترتيب (على سبيل المثال، باستخدام خوارزمية التعمية RSA). يمكن للمستخدم أن ينسخ المفتاح العام (id_rsa.pub) إلى الجهاز البعيد في المجلد ssh. للمستخدم البعيد، في ملف المفاتيح المصرح لها authorized_keys. يمكن أن يحوي هذا الملف أي قدر من المفاتيح العامة، كأماكن يُرغب بعمل اتصال بعيد بالجهاز منها. الصيغة هي مفتاح واحد في كل سطر، وهو مكافئ للملف rhosts. (لكن السطور لها حجم مختلف). بعد إدخال المفاتيح العامة للمستخدم-الجهاز في هذا الملف، فسيتمكن المستخدم من الاتصال من ذاك الجهاز دون الحاجة لكلمة مرور.

في الوضع العادي (دون إنشاء المفاتيح)، فسيتم طلب كلمة مرور من المستخدم، ولكن الاتصال سيكون دائماً معمّىً، ولن يكون بإمكان مستخدمين آخرين - يمكن أن يكونوا يراقبون الشبكة - أن يصلوا إليه. لمعلومات إضافية، انظر في man ssh. لتنفيذ أمر عن بعد، ببساطة [نفّذ]:

ssh -l *login name host_remote_command*

على سبيل المثال:

ssh -l user localhost ls -a

3.2.2 sshd

إن sshd هو خادم (مراقب) ssh (إذا لم يكن مثبتاً، فيمكن تثبيته بالأمر apt-get install ssh) الذي سيثبت الخادم والعميل). إضافة إلى ذلك، فهذا الأمر يستبدل rlogin و telnet و rsh ويقدم اتصالاً آمناً ومعمّىً بين مضيفين غير آمنين في الشبكة.

وبشكل عام سيبدأ هذا بملف الابتداء (/etc/init.d أو /etc/rc) وانتظار الاتصال من العملاء. إن sshd في معظم التوزيعات الحالية يدعم الإصدارين الأول والثاني من ميفاق ssh. عند تثبيت الحزمة تُنشئ مفتاح RSA محدّد للمضيف، وعند تشغيل المراقب تُنشئ واحداً آخر، وهو RSA للجلسة، والذي لا يتم حفظة في القرص ويتغير كل ساعة. عندما يبدأ عميل اتصالاً، يُنشئ العميل رقماً عشوائيًّا مكوناً من 256 خانة ثُنائيّة يتم تعميته بمفتاحي الخادم معاً ثم إرساله إليه. سيتم استخدام هذا الرقم أثناء الاتصال كمفتاح للجلسة لتعمية الاتصال باستخدام خوارزمية تعمية معيارية. يمكن للمستخدم اختيار أيّ من الخوارزميات المتاحة التي يقدمها الخادم. هناك اختلافات (أكثر أمناً) عند استخدام الإصدار الثاني من الميفاق. ومن ثم تتم بعض طرق استيثاق المستخدم المذكورة في العميل أو سيتم السؤال عن كلمة المرور، ولكن مع كون الاتصال معمىً دائماً. لمعلومات إضافية، راجع man ssh.

3.2.3 التمرير عبر ssh

كثيراً ما يكون لدينا وصول إلى خادم sshd ولكن – لأسباب أمنية – لا يكون لدينا وصول إلى خدمات أخرى غير معماة (كخدمة البريد POP3 أو خادم النوافذ X11 على سبيل المثال) أو ببساطة نرغب بالاتصال بخدمة يمكن الوصول إليها من بيئة الشركة فقط. لعمل ذلك، يمكن عمل قناة اتصال معمّاة tunnel بين جهاز العميل (على سبيل المثال باستخدام نظام ويندوز وبرنامج عميل ssh حر يدعى putty) والخادم الذي يعمل عليه sshd. في هذه الحالة عندما نوصل القناة بالخدمة، فسترى الخدمة الطلب وكأنه قادم من نفس الجهاز. على سبيل المثال، إذا كنا نرغب بإنشاء اتصال POP3 على المنفذ 110 للجهاز البعيد (التي لها خادم sshd أيضاً) سننفذ:

```
ssh -C -L 1100:localhost:110 user-id@host
```

سيسأل هذا الأمر عن كلمة المرور لـ user-id@host، وبمجرد أن يتم الاتصال، سيكون قد تم إنشاء القناة. كل حزمة تُرسل إلى الجهاز المحلي على المنفذ 1100 سيتم إرسالها إلى الجهاز البعيد على المنفذ 110 حيث تستمع خدمة POP3 (يضغط الخيار C- الاتصال في القناة).

إنشاء قنوات على منافذ أخرى سهل جدًّا. لنفرض مثلاً أنه لدينا اتصال فقط إلى خادم وسيط بعيد من جهاز بعيد (ولوج عن بعد) – وليس من الجهاز نفسه –، فيمكننا إنشاء قناة لوصل المتصفح عبر القناة في الجهاز المحلي. لنفرض أن لدينا ولوجاً إلى جهاز يُدعى gateway يمكنه الوصول إلى الجهاز المدعو proxy الذي يشغل الخادم الوسيط Squid على المنفذ 3128. ننفذ:

```
ssh -C -L 8080:proxy:3128 user@gateway
```

بمجرد أن يتم الاتصال، فسيكون لدينا قناة تستمع عبر المنفذ المحلي 8080 لتقوم بتحويل سير البيانات من gateway إلى proxy إلى المنفذ 3128. للتصفح بأمان، كل ما نحتاج لعمله هو http://localhost:8080 [4].

4 يقصد هنا إضافة هذا العنوان إلى إعدادات البرنامج (المتصفح في هذه الحالة) كخادم وسيط ليتم تمرير الاتصال عبره ومن ثم إلى القناة المعماة التي أنشأناها.

4 خدمات نقل الملفات: FTP

ميفاق نقل الملفات FTP هو ميفاق عميل/خادم (تحت TCP) يسمح بنقل الملفات من وإلى نظام بعيد. إن خادم ftp هو حاسوب يشغّل المراقب ftpd[5].

4.1 عميل ftp (متعارف عليه)

يسمح عميل FTP بالوصول إلى خوادم FTP وهناك عدد هائل من عملاء FTP المتاحين. استخدام FTP سهل جدًّا، من سطر الأوامر، نفّذ:

```
ftp server-name
```

أو ftp ومن ثم بالتفاعل معه:

```
open server-name
```

سيطلب الخادم اسم المستخدم وكلمة المرور (إذا كان يقبل المستخدمين المجهولين anonymous، فسنستخدم anonymous كاسم المستخدم وبريدنا ككلمة المرور)، ومن سطر الأوامر(باتباع عدّة رسائل) سنكون قادرين على بدء نقل الملفات. يسمح الميفاق بالنقل في الوضعين ASCII و binary. من المهم أن نقرر أي نوع من الملفات سيتم نقله لأن نقل ملف ثنائي في وضع ASCII سيُتلف الملف. للتغيير بين الوضعين، سيكون علينا تنفيذ الامر ascii أو binary. من الأوامر المفيدة لعميل FTP الأمر ls (تصفح المجلد البعيد)، و get file_name (لتنزيل الملفات) أو mget، و put file_name (لإرسال ملفات إلى الخادم) أو mput؛ نحتاج في هاتين الحالتين الأخيرتين لأن نكون مخولين للكتابة في مجلد الخادم. يمكننا تنفيذ أوامر محلية بإدخال "!" قبل الأمر. على سبيل المثال، cd /tmp! يعني بأن الملف الذي يتم تنزيله إلى الجهاز المحلي سيتم تنزيله إلى المجلد /tmp/. من أجل عرض حالة وعمل النقل، سيكون العميل قادراً على طباعة علامات، أو إشارات يتم تفعيلها بالأمرين hash و tick. هناك أوامر أخرى يمكن الرجوع إليها في صفحة الدليل (man ftp) أو بتشغيل المساعدة من داخل العميل[6].

---

5 من خوادم ftp الشائعة في لينكس vsftpd و lftpd.

6 يمكن طلب المساعدة بتنفيذ الأمر help في عميل ftp.

لدينا بدائل عديدة للعملاء، فمثلاً لدينا في سطر الأوامر: ncftp, lukemftp, lftp, cftp, yafc، وفي الوضع الرسومي: gFTP, WXftp, LLNL XFTP, guiftp.[7]

4.2 خوادم ftp

يعمل خادم يونكس التقليدي عبر المنفذ 21 ويتم تشغيله عبر المراقب inetd (أو xinetd، وهذا يعتمد على أيهما المثبت). في inetd.conf، يُنصح بتضمين tcpd wrapper وقواعد الوصول في host.allow و host.deny في الاتصال إلى ftpd من inetd لزيادة أمن النظام (راجع الجزء المتعلق بالأمن). عندما تستقبل اتصالاً، تقوم بالتأكد من اسم المستخدم وكلمة المرور وتسمح بالدخول إذا كان الاستيثاق صحيحاً. يعمل FTP المجهول (anonymous) بشكل مختلف، حيث سيتمكن المستخدم من الوصول فقط إلى مجلد تم تضمينه في ملف الإعداد والشجرة المتفرعة عنه، ولكن ليس ما يعلوه، وذلك لأسباب أمنية. وبشكل عام يحوي المجلد المجلدات pub و bin و etc و lib بحيث يتمكن مراقب FTP من تنفيذ أوامر خارجية لطلبات ls. يدعم مراقب FTP الملفات التالية لإعداده:

- /etc/ftpusers: قائمة المستخدمين غير المقبولين في النظام، مستخدم في كل سطر.
- /etc/ftpchroot: قائمة المستخدمين الذين سيتم تغيير مجلد chroot لهم عند اتصالهم. ضروري عندما نرغب بضبط خادم للمجهولين.
- /etc/ftpwelcome: رسالة الترحيب.
- /etc/motd: الأخبار بعد الولوج.
- /etc/nologin: رسائل تظهر عند رفض الاتصال.

---

7 لدينا أيضاً filezilla وغيره، إضافة إلى أن معظم المتصفحات ومدراء الملفات يدعمون FTP إما مباشرة أو باستخدام إضافات، وبعضها يدعم الكتابة (أي إرسال الملفات والتعديل على ما في الخادم)، والبعض الآخر يسمح بالقراءة (أي تنزيل الملفات) فقط.

- /var/log/ftpd: سجلّ عمليات النقل.

إذا رغبنا في مرحلة ما أن نمنع اتصال FTP، فيمكننا عمل ذلك عبر تضمين الملف /etc/nologin. سيعرض ftp محتوياته وينتهي. إذا كان هناك ملف message. في مجلد فسيعرضه ftpd عند الوصول إليه.

يمر اتصال المستخدم عبر 5 مستويات مختلفة:

١) وجود كلمة مرور صالحة.

٢) عدم ظهوره في القائمة /etc/ftpusers.

٣) وجود صدفة معيارية صالحة.

٤) إذا كان موجوداً في /etc/ftpchroot فسيتم الانتقال إلى مجلد المنزل (يتم تضمينه إذا كان Anonymous أو FTP).

٥) إذا كان المستخدم anonymous أو FTP، فيجب أن يكون له مدخلة في /etc/passwd مع عميل FTP، ولكن سيظل من الممكن أن يتصل بإعطاء أي كلمة مرور (المتعارف عليه استخدام عنوان البريد الإلكتروني).

علينا أن نبقي في بالنا أن المستخدمين المصرح لهم باستخدام FTP فقط ليس لديهم صدفة في مدخلة المستخدم ذات العلاقة في /etc/passwd لمنع هذا المستخدم من الحصول على اتصال عبر telnet أو ssh على سبيل المثال. ولهذا، فعند إنشاء مستخدم، فسيكون علينا أن نشير – على سبيل المثال – :

```
useradd -d nteum -s /bin/false nteum
```

ومن ثم:

```
passwd nteum
```

والتي تعني بأنه لن يكون للمستخدم nteum صدفة للاتصال التفاعلي (إذا كان المستخدم موجوداً مسبقاً، فيمكننا تحرير الملف /etc/passwd وتغيير الحقل الأخير إلى /bin/false). ثم سيكون علينا إضافة /bin/false كسطر أخير في /etc/shells. يشرح المرجع [Mou01] خطوة بخطوة كيفية إنشاء خادم FTP آمن بمستخدمين مسجّلين وخادم FTP

للمجهولين للمستخدين غير المسجلين. إن WUFTPD (الموقع http://www.wuftpd.org) و ProFTPD (الموقع http://www.proftpd.org) إثنان من أكثر خوادم FTP غير المعيارية شيوعاً.

لتثبيت ProFTPD في دبيان نفّذ: apt-get install proftpd. بعد تثبيته، سيسأل debconf إذا كنت ترغب بتشغيله عبر inetd أو بالوضع اليدويّ (ينُصح باستخدام الطريقة الثانية). إذا كنّا نرغب بإيقاف الخدمة (لتغيير الإعداد مثلاً)، يمكننا أن نستخدم /etc/init.d/proftpd stop، ولتغيير الملف يمكننا استخدام /etc/proftpd.conf.

راجع http://www.debian-administration.org/articles/228 لضبطه في الوضع الآمن (TSL) أو للحصول على وصول للمجهولين.

من خوادم دبيان المثيرة للاهتمام بشدّة PureFtpd (الحزمة pure-ftpd) الآمنة جدًّا، والتي تسمح بوجود مستخدمين افتراضيين، وتقسيم للحصص، و SSL/TSL، ومجموعة من المزايا التي تستحق الاهتمام. يمكننا تفحص تثبيته/ضبطه في http://www.debian-administration.org/articles/383.

5 خدمات تبادل المعلومات على مستوى المستخدم

5.1 وكيل نقل البريد MTA

وكيل نقل البريد Mail Transport Agent – MTA مسؤول عن إرسال/استقبال البريد من خادم بريد إلكتروني إلى/من الإنترنت باستخدام ميفاق نقل البريد البسيط Simple Mail Transfer Protocol – SMTP. دبيان تستخدم exim بشكل مبدئي، وذلك لأنه أسهل للضبط من حزم MTA الأخرى، مثل smail أو sendmail (الثاني أحد أسلافه). يقدم exim مزايا متقدمة مثل رفض الاتصال من مواقع السُخام المعروفة، وبه دفاعات ضد البريد غير المرغوب وتفجير البريد[8]، وهي فعالة جدًّا في معالجة كمّ كبير من رسائل البريد. يتم تشغيلها عبر inetd بسطر في ملف الإعداد /etc/inetd.conf بمعاملات للإعداد العاديّ (أو xinetd).

يستخدم exim ملف الإعداد /etc/exim/exim.conf الذي يمكن تعديله يدويًّا، ولكن ينصح بعمل التعديلات باستخدام نصّ برمجيّ يدعى eximconfig، وذلك للتمكن من ضبط exim تفاعليًّا. ستعتمد قيم الإعداد على وضع الجهاز، رغم هذا فوصلها سهل جدًّا، حيث يقترح النصّ البرمجي القيم المبدئيّة. إضافة إلى ذلك، يمكننا أن نجد في /usr/doc/exim أمثلة على إعدادات اعتيادية.

يمكننا أن نتفحص ما إذا كان الإعداد صالحاً عبر exim-bV، فإذا كانت هناك أخطاء في ملف الإعداد فسيعرضها البرنامج على الشاشة، أما إذا كان كل شيء صحيحاً، فستعرض الإصدار والتاريخ. لاختبار ما إذا كان يتعرف على صندوق بريد محلي، استخدم:

```
exim -v -bt local_user
```

التي ستعرض طبقات النقل المستخدمة والعنوان المحلي للمستخدم. يمكننا أيضاً أن نقوم بالاختبار التالي مع مستخدم بعيد بتبديل local_user إلى مستخدم بعيد لديه عنوان بعيد لرؤية سلوكه. ثم جرب إرسال رسالة بريد محلياً وخارجيًّا، بتمرير

---

8 إن تفجير البريد أو mail bombing هي تقنية تعتمد على إغراق عنوان بريد إلكتروني معين أو مجموعة من العناوين بكمّ هائل من الرسائل وذلك لجعل الوصول إلى الرسائل الهامة صعباً أو متعذّراً أو لتعطيل الخدمة بتجاوز الحصة المسموح بها للمستخدم.

الرسالة مباشرة إلى exim (دون استخدام عميل، على سبيل المثال، mailx)، بوضع (كل هذه المعلومات معاً) مثلاً:

```
exim postmaster@OurDomain
From: user@domain
To: postmaster@OurDomain
Subject: Test Exim
Test message
^D
```

ثم يمكننا أن نحلل ملفي التتبع mainlog و paniclog في /var/log/exim/ لنرى سلوكه ولرؤية أيّة رسائل خطأ تم إنشاؤها. ومن البديهي أنه يمكننا أيضاً الاتصال بالنظام بحساب المستخدم postmaster (أو المستخدم الذي تم إرسال البريد إليه) وقراءة رسائل البريد لرؤية ما إذا كان كل شيء على ما يرام. تتم الطريقة الأخرى عبر تشغيله في وضع التصحيح باستخدام dNro- كمعامل، حيث Nro هو مستوى التصحيح (1-9). المعامل العادي الذي علينا تشغيلها به هو exim -bs، سواء عبر inetd أو xinetd. يمكننا أيضاً تشغيله كمعالج عبر /etc/init.d/exim start في الأنظمة التي تطلب إمكانات عالية لمعالجة البريد. انظر إلى التوثيق (مضمنة في ديبان في الحزمة exim-doc-html) من أجل ضبط المُرَشِّحات، والتحقق من المضيفات، والمرسِلين، إلخ. من المفيد أيضاً تثبيت الحزمة eximon، وهي مراقب exim الذي يسمح للمدير برؤية طابور رسائل البريد والطوابير والتصرف بالرسائل التي في الطابور من أجل توزيعها (التجميد والردّ والرمي...).

الإصدار الأخير من exim هو exim4 (يمكن تثبيته بالأمر apt-get install exim-daemon-heavy، ثبّت أيضاً exim4-config الذي يساعد في ضبط exim4) – أبق في بالك أن هناك حزماً مختلفة بإمكانيات مختلفة لكن exim-daemon-heavy هو أكثرها كمالاً. ننصح بقراءة /usr/share/doc/exim/README.Debian.gz و http://www.exim.org/docs.html. إحدى الاختلافات الصغيرة التي علينا مراعاتها في الإعداد أنه بدلاً من وجود ملف إعداد واحد وهو exim.conf (الخيار المبدئي إذا ثبتنا exim من المصدر) تستخدم الحزمة exim4-config (نُصح بتثبيتها) ملفات إعداد صغيرة بدلاً من ملف واحد، وتكون تلك الملفات في /etc/exim4/conf.d/ ويتم تضمينها في ملف واحد (ويكون مبدئيًّا /var/lib/exim4/config.autogenerated) بالأمر update-exim4.conf

5.2 ميفاق الوصول لرسائل الإنترنت (IMAP)

تسمح هذه الخدمة بالوصول إلى رسائل البريد المخزنة في خادم وحيد عبر عميل بريد مثل thunderbird أو seamonkey (كلاهما في mozilla.org) تسمح هذه الخدمة بمساعدة المراقب imapd (الإصدارات الحالية تدعم الميفاق IMAP4rev1) بوجود ملف بريد إلكتروني على جهاز بعيد. تتوفر خدمة imapd عبر المنفذ 143 (imap2) أو 993 (عندما تكون التعمية عبر SSL مدعومة) (imaps). إذا استخدمنا inetd، يتم تشغيل هذا الخادم عبر سطر في /etc/inetd.conf مثل:

```
imap2 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
imap3 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
```

في هذا المثال، يُستدعى tcpd wrapper الذي يعمل مع hosts.allow و hosts.deny من أجل زيادة الحماية. أشهر التطبيقات uw-imapd (أو "جامعة واشنطن" University of Washington، وهو مثبت مبدئيًّا في دبيان) أو إصدارها الآمن uw-imapd-ssl، ولكن لدينا أيضاً cyrus-imap أو courier-imap. لاختبار ما إذا كان خادم imap يعمل، علينا أن نستخدم عميلاً، مثل seamonkey-mail وإنشاء حساب لمستخدم محلي وضبطه بشكل مناسب بحيث يتصل عبر الجهاز المحلي ، للتحقق من أن imap يعمل بشكل صحيح.

إصدار imap في دبيان تم تصريفه ليدعم MD5 كطريقة استيثاق للمستخدمين البعيدين، لتعمية كلمات مرور الاتصال، ولتجنب الهويات المستبدلة عبر التنصّت على الشبكة (يجب أن يكون العميل المتصل بخادم imap داعماً لطريقة الاستيثاق عبر MD5 أيضاً). الطريقة بسيطة جدًّا وآمنة، ولكن يجب على الخادم أن يعلم كلمات المرور لمستخدمي البريد كنصّ صرف، مما يعني أنه ينصح باستخدام إصدار imap على SSL الذي يعمل عبر المنفذ 993. مثل ssh، يعمل الميفاق imap بتعمية الاتصال باستخدام شهادة المضيف (يجب أن يدعم العميل المتصل بالخادم طريقة الاتصال هذه، على سبيل المثال thunderbird و seamonkey -mail). لضبط الخادم imaps، ثبّت الحزمة الدبيانية uw-imapd-ssl وهي خادم imap بدعم ssl.

سيُنشئ التثبيت شهادة موقعة ذاتياً صالحة لعام واحد ومخزنة في /etc/ssl/certs/imapd.pem. يمكن استبدال هذه

الشهادة بواحدة موقّعة من شركة معتمدة أو يمكن إنشاء واحدة خاصة باستخدام OpenSSL. يُنصح بترك مدخلة imaps فقط في الملف /etc/inetd.conf وإزالة مدخلتي imap2 و imap3 إذا كنّا نرغب بأن يتم الوصول إلى imap عبر SSL فقط.

وهناك ميفاق آخر بمزايا مشابهة كان شائعاً جدًّا في الماضي، لكن احتل مكانه IMAP الآن، وهو ميفاق مكتب البريد Post Office Protocol – POP الإصدار 2 و 3. يتم تثبيته وتشغيله بنفس طريقة IMAP. هناك العديد من خوادم POP، لكن أشهرها courier-pop، و cyrus-pop3d، و ipopd (جامعة واشنطن)، و qpopper، و solid-pop3d.

5.2.1 نواحٍ تكميليّة

لنفرض أننا كمستخدمين لدينا 4 حسابات بريد على خوادم مختلفة وأننا نرغب بتجميع كل الرسائل المرسلة إلى كل هذه الحسابات إلى حساب واحد، أن نصل إلى ذلك الحساب خارجيًّا، وأن يحوي أيضاً على مرشحات للسخام.

أولاً، سيكون علينا تثبيت exim و IMAP والتحقق من أنهما يعملان. علينا أن نأخذ بعين الاعتبار أننا إذا ثبتنا courier-imap (الذي يقول عنه بعض المؤلفين أنه أفضل من wu-imapd) فسيعمل طبقاً لصيغة بريد تدعى Maildir، وأنه سيكون علينا أن نضبط exim أيضاً ليعمل على maildir بالإعداد التالي في /etc/exim/exim.conf (أو الملف ذي العلاقة إذا كان لدينا exim4)، بتغيير قيمة الخيار mail_dir format إلى true (سيتم تخزين البريد في حساب للمستخدم المحلي في مجلد يُدعى Maildir). ثم سيكون علينا إعادة بدء الخادم exim بالأمر /etc/init.d/exim restart، وإعادة الاختبار التنفيذي بإرسال رسالة بريد لنا وقراءتها بعميل يدعم maildir (على سبيل المثال mutt -mailx لا يدعمها – راجع http://www.mutt.org).

سنستخدم fetchmail لاستخلاص البريد من الحسابات المختلفة (الذي يمكن تثبيته بالأمر apt-get install fetchmail). ومن ثم سيكون علينا إنشاء الملف fetchmailrc. في مجلد المنزل الخاص بنا (يمكننا أيضاً أن نستخدم الأداة fetchmailconf) التي يفترض أنها تحوي شيئاً مثل:

```
set postmaster "pirulo"
set bouncemail
set no spambounce
set flush

poll pop.domain.com proto pop3
user 'user1' there with password 'secret' is pirulo here

poll mail.domain2.com
user 'user5' there with password 'secret2' is 'pirulo' here
user 'user7' there with password 'secret3' is 'pirulo' here
```

تخبر مجموعة الأعمال Fetchmail أن هذا السطر يحوي خياراً عامًّا (إرسال الأخطاء، وحذف البريد من الخوادم ...).

ومن ثم، سنحدد خوادم البريد: واحد للتحقق مما إذا كان هناك بريد بالميفاق POP3، وآخر لاختبار استخدام العديد من الموافيق لإيجاد واحد يعمل. نفحص بريد شخصين بخيار الخادم الثاني، لكن يتم إرسال كل البريد الذي يُعثر عليه إلى مجمع بريد pirulo. يسمح لنا هذا بتفقد العديد من صناديق البريد لخوادم مختلفة كما لو كانت صندوق MUA وحيد. إن المعلومات المحددة لكل مستخدم تبدأ بالمستخدم المنفّذ. يمكن أن يوضع fetchmail في cron (على سبيل المثال في /var/spool/cron/crontabs/pirulo بإضافة 1 * * * * /usr/bin/fetchmail -s)، بحيث يعمل تلقائيًّا، أو يمكن تشغيله في وضع المراقب (ضع set daemon 60 في fetchmailrc. (وشغله مرة على سبيل المثال في Autostart لجنوم/كدي أو في bashrc. – ستعمل كل 60 ثانية).

لإزالة البريد غير المرغوب به سنستخدم SpamAssassin (نثبته بالأمر apt-get install spamassassin) ويمكننا أن نضبط Kmail أو Evolution (تفحص المراجع لمعرفة طريقة ضبطها) لكي تشغّله. في هذا الإعداد سنستخدم Procmail، وهي أداة قوية جدًّا (تسمح بتوزيع البريد، والترشيح، وإعادة الإرسال آليًّا...). بمجرد تثبيته (بالأمر apt-get install procmail)، فسنحتاج لإنشاء ملف يُدعى procmailrc. في منزل كل مستخدم لاستدعاء Spamassassin:

- اجعل قيمتها "yes" للرسائل الوظيفية ورسائل التصحيح:

```
VERBOSE = no
```

- نفترض بأن الرسائل في Maildir./~، غيرها إذا كانت غير ذلك

```
PATH=/usr/bin:/bin:/usr/local/bin:
MAILDIR=$HOME/Maildir
DEFAULT=$MAILDIR/
# Directory for storing the files
PMDIR=$HOME/.procmail
# Comment if we do not want a log of Procmail
LOGFILE=$PMDIR/log
# Spam filter
INCLUDERC=$PMDIR/spam.r
```

يحوي الملف ~/.procmail/spam.rc

```
# If the spamassassin is not on the PATH
# add the directory to the PATH variable:
# 0fw: spamassassin.lock|
| spamassassin - a

# The three following lines will move
# Spam mail to a directory called
# "spam-folder". If we want to save it in the Inbox, so that
# it can be filtered later with the client, comment the three lines.

:0:
* ^X-Spam-Status: Yes
spam-folder
```

يحوي الملف ~/.spamassassin/user_prefs بعض الإعدادات المفيدة لـ spamassassin (انظر إلى المراجع):

```
#User preferences file. Ver man
#Mail::SpamAssassin::Conf
#Threshold for recognising a Spam: #Default 5, but with 4 it works a bit better
required_hits 4
# Sites we will never consider Spam to
#come from
whitelist_from root @debian.org
whitelist_from *@uoc.edu
#Sites SPAM always comes from
#(separated by commas)
blacklist_from viagra@domain.com
#Addresses on Whitelist and blacklist are
#global patterns such
#as:"friend@place.com", "*@isp.net", or
#"*.domain.com".
#Insert the word "[SPAM]" in the subject
#(to make filtering easier).
#If we do not wish to comment the line.
subject_tag [SPAM]
```

سيُنشئ هذا وسم X-Spam-Status: yes في ترويسة الرسالة إذا كانت تعتقد بأن الرسالة Spam. ثم سيكون علينا أن نرشحها ونضعها في ملف آخر أو نحذفها مباشرة. يمكننا أن نستخدم procmail لترشيح البريد من النطاقات والمستخدمين، إلخ. لمعلومات إضافية، زُر http://www.debian-administration.org/articles/242. وفي النهاية، يمكننا أن نثبت عميل بريد وأن نضبط المرشحات بحيث تحدد كل رسائل البريد عبر X-Spam-Status: yes، وتحذفها أو ترسلها إلى مجلد نحقق فيه فيما بعد من القرارات الإيجابية الخاطئة (الرسائل التي تم التعرف عليها على أنها junk مع أنها ليست كذلك). ومن النواحي التكميلية لهذا التثبيت أنه إذا كنا نرغب بالحصول على خادم بريد عبر webmail (وبعبارة أخرى، للتمكن من تفحص البريد من الخادم عبر متصفح دون الحاجة لتثبيت أو ضبط عميل - كمراجعة حسابات gmail و hotmail)، فمن الممكن تثبيت Squirrelmail (عبر apt-get install squirrelmail) من أجل تقديم هذه الخدمة. لدبيان، زُر http://www.debian-administration.org/articles/200.

هناك إمكانات أخرى كما ذُكر في http://www.debian-administration.org/articles/364، كتثبيت MailDrop بدلاً من Procmail، أو postfix بدلاً من Exim، أو تضمين Clamav/Amavisd كمضاد للفيروسات (يسمح

Amavisd لـ postfix بأن يتم وصله بكل من spamassassin و clamav).

5.3 الأخبار

الأخبار أو مجموعات النقاش مدعومة عبر ميفاق نقل الأخبار عبر الشبكة – Network News Transfer Protocol NNTP. تثبيت خادم أخبار ضروري إذا كنا نرغب بقراءة الأخبار دون اتصال، أو إذا كنا نرغب بإيجاد مُعيد للخوادم المركزية، أو إذا كنا نرغب بإيجاد خادم أخبار رئيسي خاص بنا. الخوادم الأكثر شيوعاً هي INN و CNEWS، ولكنها حزم مركبة مصممة للخوادم الكبيرة. إن Leafnode حزمة USENET تستخدم خادم TNP، مطوعة خصيصاً للمواقع ذات المجموعات الصغيرة من المستخدمين والتي نرغب بالوصول منها إلى عدد كبير من مجموعات الأخبار. هذا الخادم مثبت بإعداد دبيان الأساسي ويمكن إعادة ضبطه بالأمر dpkg-reconfigure leafnode لكل المعاملات كالخوادم المركزية، ونوع الاتصال، إلخ. يبدأ هذا المراقب من inetd بطريقة مشابهة لـ imap (أو بـ xinetd). يدعم leafnode مرشحات عبر تعابير اعتيادية regular expressions[9] (من النوع ^Newsgroups:. * [,] alt.flame$) في /etc/news/leafnode/filters، الذي تكون فيه الترويسة لكل رسالة مقارنة بالتعبير الاعتيادي، وإذا كان هناك مطابقة، فسيتم رفض الرسالة.

الخادم يسهل ضبطه، ويجب أن تكون كل الملفات ملكاً لمستخدم أخبار بتخويل للكتابة (تأكد من وجود هذا المالك في /etc/passwd. كل ملفات التحكم والأخبار والإعداد يمكن إيجادها في /var/spool/news، عدا إعداد الخادم نفسه الموجود في الملف /etc/news/leafnode/config. للملف الإعداد معاملات ضرورية يجب ضبطها (على سبيل المثال، ليتمكن الخادم من الاتصال بالخوادم الرئيسية). إنها server (خادم إخبار سيتم الحصول على الرسائل منه وإرسالها إليه) و expire (عدد الأيام التي سيتم حذف الجلسة أو الموضوع المقروء بعدها). وكذلك الأمر، علينا أن نضبط معاملات اختيارية بطبيعة عامة أو

9 التعابير الاعتيادية، Regular expressions، أو RE أو Regex، هي طرق للتعبير المنطقي عن صيغة معينة لمحتوى، كأن تقول بأن النص يجب أن يتكون من أرقام فقط، أو من القيم كذا وكذا مكررة للعدد الفلاني من المرات، أو مكررة بشكل لا نهائي، أو أن يتكون من 5 حروق صغيرة مثلاً، أو أن القيمة تبدأ بأحد الرموز التالية أو تنتهي بالرمز الفلاني ...إلخ. يمكن مراجعة الموضوع ذي العلاقة في "كتاب لينكس الشامل" للأستاذ مؤيد السعدي لمزيد من المعلومات حول التعابير الاعتيادية.

محددة يمكن ضبطها. لمعلومات إضافية، انظر إلى التوثيق (عبر man leafnode أو /usr/doc/leafnode/README.Debian).

لفحص أداء الخادم، يمكننا تشغيل:

```
telnet localhost nntp
```

وإذا كان كل شيء يعمل بشكل صحيح، فستظهر معرف الخادم وستنتظر أمراً، وكاختبار يمكننا إدخال help [للإلغاء نضغط Ctrl + C (ثم Quit للخروج)].

5.4 الشبكة العنكبوتية العالمية و httpd

أباتشي Apache أحد أكثر الخوادم شعبية بأفضل الإمكانيات فيما يتعلق بميفاق نقل النصوص التشعبية hyper text transfer protocol – http. لأباتشي تصميم مقسم إلى وحدات، ويدعم امتدادات الوحدات الديناميكية أثناء تشغيله. وهو قابل للتطويع بشكل كبير من ناحية عدد الخوادم والوحدات المتاحة، ويدعم آليات عديدة للاستيثاق والتحكم بالوصول وملفات المعلومات metafiles، والاختزان المؤقت للوسيط، والخوادم الافتراضية، إلخ. وباستخدام الوحدات (المضمنة في دبيان) يمكن الحصول على PHP, Perl, Java Servlets, SSL وامتدادات أخرى (انظر إلى التوثيق في http://www.apache.org).

أباتشي مصمم ليتم تشغيله كعملية مراقب مستقل. بهذه الطريقة يُنشئ مجموعات من العمليات الثانوية التي ستتعامل مع طلبات الإدخال. يمكن أيضاً تشغيلها كمراقبات إنترنت عبر inetd، مما يعني أنها ستعمل في كل مرة تتلقى فيها اتصالاً. يمكن أن يكون إعداد الخادم معقداً جداً وهذا يعتمد على المتطلبات (تفحص التوثيق)، ولكن يمكننا أن نرى هنا الحد الأدنى من الإعداد المقبول. ملفات الإعداد موجودة في /etc/apache، وهي httpd.conf (ملف الإعداد الرئيسي)، و srm.conf، و access.conf (الاثنان الأخيران باقيان للتوافقية)، و mime.conf (صيغ MIME)، و magic (رقم التعريف). ملفات التقارير موجودة في /var/log/apache، وهي error.log (تسجل الأخطاء في طلبات الخادم)، و access.log (سجل بمن قاموا بالوصول وإلى ماذا)، و apache.pid (معرف العملية).

يبدأ أباتشي من النص البرمجي للتشغيل /etc/init.d/apache و /etc/rcX.d/، لكن يمكن التحكم به يدوياً عبر الأمر apachectl. يمكن للأمر apacheconfig أيضاً أن يُستخدم من أجل ضبط الخادم. المجلدات المبدئية في ديبان هي:

- /var/www/ : مجلد للملفات HTML.
- /usr/bin/cgibin/ : مجلد للملفات التنفيذية cgi على الخادم.
- http://server.domain/user : الصفحة الشخصية للمستخدم user.
- /home/~user/public.html: مجلد للصفحات الشخصية.

الملف المبدئي المقروء من كل مجلد هو index.html. بعد تثبيت الحزمتين apache و apache-common، فإن ديبان ببساطة تضبط الخادم وتبدأه. يمكننا التحقق من أنه يعمل بفتح متصفح ما (مثل كونكيرور) وكتابة http://localhost في شريط العنوان الذي سيحمل الصفحة /var/www/index.html.

5.4.1 إعداد آلي (بالحد الأدنى) للملف httpd.conf

لنلق نظرة على أهم المعاملات التي يمكن ضبطها في أباتشي (المثال مأخوذ من الإصدار الأول من أباتشي، وهناك تغييرات طفيفة إذا كنّا نستخدم الإصدار الثاني منه).

ServerType standalone	مستحسن، أكثر فاعلية
ServerRoot /etc/apache	مكان وجود ملفات الإعداد
Port 80	مكان استماع الخادم للطلبات
User www-data Group www-data	المستخدم و المجموعة الذان سيشغّل الخادم بهما (مهمان للأمن) ويجب أن يكون كلاهما صحيحاً (يمكن أن يكونا مقفلين).
ServerAdmin webmaster@pirulo.remix.com	عنوان المستخدم الذي سيستقبل رسائل الخطأ
ServerName pirulo.remix.com	اسم الخادم المرسل إلى المستخدمين – يجب أن يكون اسماً صحيحاً وموجوداً في hosts أو DNS –
DocumentRoot /var/www	المجلد الذي ستكون فيه المستندات
Alias /icons/ /usr/share/apache/icons/	مكان وجود الأيقونات
ScriptAlias /cgibin/ /usr/lib/cgibin	مكان وجود النصوص البرمجية من نوع CGI

5.4.2 أباتشي 2,2 + **MySQL** + **PHP** + **SSL**

من النواحي المهمة في خوادم الوب الديناميكية هي الاستفادة قدر الإمكان من المزايا التي يوفرها أباتشي في الوضع الآمن SSL، و PHP (وهي وهي لغة برمجة تستخدم عموماً لإنشاء محتوى مواقع وب)، و MySQL + PHPMyAdmin (قاعدة بيانات سيتم مناقشتها في الأجزاء اللاحقة وواجهة رسومية لإدارتها) تعمل كلها معاً. سنبدأ بتثبيتها في دبيان، لكن ليس عبر حزم دبيان، بل من البرمجية التي سننزلها من المواقع الخاصة بكل منها، وبهذا يمكننا تكرار العمل على التوزيعات الأخرى. ومن البديهي أننا لن نتمكن من إدارة هذه الحزم باستخدام apt أو برنامج إدارة حزم آخر. علينا أن ننتبه للإصدارات التي يمكن أن تتغير، وأن لا نثبت الحزمة فوق حزمة مثبتة مسبقاً.

أ- تنزيل الملفات الضرورية (في المجلد /root/ على سبيل المثال، وننتقل إليه بالأمر cd /root):

١. أباتشي: من http://httpd.apache.org/download.cgi، الملف httpd-2.2.4.tar.bz2

٢. PHP: من http://www.php.net/downloads.php، الملف PHP 5.2.1 من النوع tar.bz2

٣. MySQL: من http://mysql.org/get/Downloads/MySQL-4.1/mysql-standard-4.1.21-pc-linux-gnu-i686.tar.gz/from/pick.

٤. PHPMyAdmin: من http://prdownloads.sourceforge.net/phpmyadmin/phpMyAdmin-2.9.1-all-languages.tar.bz2?download.

ب- أدوات أخرى: bzip2 libssl-dev openssl gcc g++ cpp make. تأكد إذا كانت مثبتة، إذا لم تكن مثبتة نفّذ:

```
apt-get install bzip2 libssl-dev openssl gcc g++ cpp make
```

ت- أباتشي:

```
cd /root
tar jvxf httpd-2،2،4،tar.bz2
cd httpd-2،2،4
```

نحدد في الخيار prefix أننا سنثبت في /usr/local/apache2 مثلاً

```
./configure --prefix=/usr/local/apache2 –with ssl=/usr/include/openssl --enable-ssl
make
make install
```

نعدل ملف الإعداد /usr/local/apache2/conf/httpd.conf وغير المستخدم والمجموعة إلى www-data:

```
User www-data
Groupe www-data
```

نغير المستخدم والمجموعة إلى مجلد البيانات:

```
chown -R www-data:www-data /usr/local/apache2/htdocs
```

نعدل على المستخدم www-data لنغير مجلد منزله في /etc/passwd:

```
www-data:x:33:33:www-data:/usr/local/apache2/htdocs:/bin/sh
```

خادم أباتشي مثبت. لتشغيله نفذ (لإيقافه غير start إلى stop):

```
/usr/local/apache2/bin/apachectl start
```

يمكننا وضع نصّ برمجي لتشغيل خادم أباتشي عند الإقلاع:

```
ln -s /usr/local/apache2/bin/apachectl /etc/rcS.d/S99apache
chmod 755 /etc/rcS.d/S99apache
```

ث- SSL:

في الملف /usr/local/apache2/conf/httpd.conf، نزيل التعليق من السطر:

```
Include conf/extra/httpd-ssl.conf
```

الملفات مُنشأة بالمفاتيح التي للخادم الآمن، في /root/ ننفّذ (عدّل الإصدارات إلى تلك التي تم تنزيلها) – (أمر ssl الأول سطر طويل ينتهي بالرقم 1024):

```
openssl genrsa -rand ../httpd-2.2.4.tar.bz2:../php-5.2.1.tar.bz2:../phpMyAdmin-2.9.1-all-languages.tar.bz2 -out server.key 1024
openssl rsa -in server.key -out server.pem
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 720 -in server.csr -signkey server.key -out server.crt
```

ننسخ الملفات ...

```
cp server.crt /usr/local/apache2/conf/
cp server.key /usr/local/apache2/conf/
```

نعيد تشغيل الخادم ...

```
/usr/local/apache2/bin/apachectl restart
```

يمكننا أن نرى كيفية إضافة وحدة SSL إلى خادم غير مثبتة فيه في http://www.debian-administration.org/articles/349.

ج- MySQL (لمزيد من المعلومات انظر إلى الجزء الثامن):

ننشئ مجموعة ومستخدماً لـ MySQL إذا لم تكن موجودة.

```
groupadd mysql
useradd -g mysql mysql
```

في المجلد الذي سنثبت فيه MySQL (وهو /usr/local/) نكتب:

```
cd /usr/local/
gunzip < /root/mysql-standard-4.1.21-pc-linux-gnu-i686.tar.gz | tar xvf - ln -s mysql-standard-4.1.21-pc-linux-gnu-i686 mysql
cd mysql
```

ننشئ قاعدة بيانات ونغير التصاريح

```
scripts/mysql_install_db --user=mysql
chown -R root .
chown -R mysql data
chgrp -R mysql .
```

يمكننا وضع نصّ برمجي لتشغيل خادم MySQL

```
ln -s /usr/local/mysql/support-files/mysql.server /etc/rcS.d/S99mysql.server
chmod 755 /etc/rcS.d/S99mysql.server
```

نشغّل الخادم

```
/etc/rcS.d/S99mysql.server start
```

يمكننا أن ندخل إلى قاعدة البيانات ونغير كلمة المرور للمستخدم الجذر لأسباب أمنية (راجع http://dev.mysql.com/doc/refman/5.0/en/index.html للصياغة)

```
/usr/local/mysql/bin/mysql
```

وفي الداخل يمكننا أن نكتب:

```
USE mysql
```

نضع كلمة السر pirulo للمستخدم الجذر

```
UPDATE user SET Password=PASSWORD('pirulo') WHERE User='root';
FLUSH privileges;
```

لندخل إلى MySQL سيكون علينا أن نكتب

```
/usr/local/mysql/bin/mysql -u root -p pirulo
```

ح- PHP (استبدل إلى الإصدارات المناسبة)

أدوات ضرورية:

```
apt-get install libxml2-dev curl libcurl3-dev libjpeg-mmx-dev zlib1g-dev libpng12-
dev
```

عندما يكون خادم أباتشي متوقفاً، يمكننا أن نكتب

```
cd /root
tar jvxf php-5،2،0.tar.bz2
cd php-5،2،0
```

يمكننا بالخيار prefix أن نحدد المكان الذي نرغب بالتثبيت فيه (الكل في سطر واحد):

```
./configure --prefix=/usr/local/php5 --enable-mbstring --with-apxs2=/usr/local/apache2/bin/apxs --with-mysql=/usr/local/mysql --with-curl=/usr/include/curl --with-jpeg-dir=/usr/include --with-zlib-dir=/usr/include --with-gd --with-xml --enable-ftp –enable-bcmath
```

```
make
make install
cp php.ini-dist /usr/local/php5/lib/php.ini
```

نعدل أباتشي (/usr/local/apache2/conf/httpd.conf) في الجزء المحدد:

```
<IfModule mime_module>
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source .phps
```

وأيضاً:

```
DirectoryIndex index.php index.html
```

ثم نشغّل الخادم.

خ- PHPAdmin

```
cd /usr/local/apache2/
```

يتم فك ضغط phpmyadmin في المجلد apache2 (كن يقظاً فيما يتعلق بأرقام الإصدارات).

```
tar jxvf /root/phpMyAdmin-2.9.1-all-languages.tar.bz2
mv phpMyAdmin-2.9.1-all-languages phpmyadmin
cd phpmyadmin
cp config.sample.inc.php config.inc.php
```

نحتاج لتعديل ملف الإعدادات (config.inc.php):

```
$cfg['blowfish_secret']='pirulo';
```

نزيل المستخدم وكلمة مرور المستخدم والتي تكون مبدئيًّا علامتي تنصيص مفردتين (') متتاليتين:

```
$cfg['Servers'][$i]['controluser'] = '';
$cfg['Servers'][$i]['controlpass'] = '';
```

نغير في أباتشي (/usr/local/apache2/conf/httpd.conf) بإضافة ما يلي ضمن <IfModule alias_module>

```
<IfModule alias_module>
        Alias /phpmyadmin "/usr/local/apache2/phpmyadmin/"
<Directory "/usr/local/apache2/phpmyadmin/">
        Order allow, deny
        Allow from all
</Directory>
```

نعيد تشغيل الخادم ويمكننا استدعاؤه عبر http://localhost/phpmyadmin

يمكن الحصول على مزيد من المعلومات من المواقع ذات العلاقة بكل تطبيق وفي LWP.

6 خدمة الوسيط : Squid

إن الخادم الوسيط Proxy Server – PS يستخدم لحفظ عرض نطاق الاتصال، ولزيادة الأمن، ولتحسين سرعة تصفح الوب.

إن سكويد أحد الخوادم الوسيطة الرئيسية، حيث أنه مفتوح المصدر وأنه يقبل ICP (خصائص تسمح بتبادل التلميحات مع الخوادم الوسيطة الأخرى)، و SSL (للاتصال الآمن بين الخوادم الوسيطة)، وتدعم عناصر FTP، و Gopher، و HTTP، و HTTPS (الآمن). عملها بسيط، فهي تحفظ العناصر التي تطلب أكثر في الذاكرة RAM، والأقل في قاعدة بيانات على القرص الصلب. يمكن أيضاً ضبط خوادم سكويد هرمياً لعمل شجرة من الوسطاء (جمع وسيط) اعتماداً على المتطلبات. هناك إعدادان ممكنان:

١) كمسرع httpd للحصول على أداء محسن لخدمة الوب.

٢) كخادم proxy-caching للسماح لمستخدمي شركة باستخدام الخادم الوسيط للخروج إلى الإنترنت.

في الوضع الأول تعمل كوسيط عكسي، وبعبارة أخرى، يستقبل طلب عميل، ثم يخدمه إذا كان طلبه موجوداً، فإن لم يكن طلبه موجوداً يطلبه ثم يمرره إلى العميل عندما يحصل عليه، ويخزنه للمرة المقبلة. في الخيار الثاني يمكن أن يستخدم كمتحكم لتقييد المواقع التي يمكن الوصول منها إلى الإنترنت، أو للسماح بالوصول في أوقات معينة من اليوم. ما إن يتم تثبيتها (الحزمة squid في دبيان، ويمكن أيضاً تثبيت squid-cgi, squidguard, squidtaild) يتم إنشاء ثلاثة ملفات: /etc/squid.conf (إعدادات)، و /etc/init.d/squid (لتشغيل الخادم)، و /etc/logrotate.d/squid (للتحكم بالتقارير).

1.1 سكويد كمسرّع للوب

في هذا الوضع، إذا كان خادم الوب على نفس الجهاز مع الخادم الوسيط، فسيكون علينا إعادة ضبطه ليستقبل الطلبات على المنفذ 81 (في أباتشي، غيّر port 80 إلى port 81 في httpd.conf). يحوي ملف الإعداد (/etc/squid.conf)

على عدد كبير من المدخلات، لكننا سنرى هنا الضرورية منها فقط:

http_port 80 icp_port 0 hierarchy_stoplist cgi-bin \? acl QUERY urlpath_regex cgi-bin \? no_cache deny QUERY	المنفذ الذي يستمع فيه لطلبات http المنفذ الذي يستمع فيه لطلبات icp
cache_mem 100 MB redirect_rewrites_host_header off cache_replacement_policy lru memory_replacement_policy lru	الذاكرة للعناصر قيد الاستخدام
cache_dir ufs /var/spool/squid 100 16 256 Database emulate_httpd_log on	نوع ومكان الاختزان على القرص
acl all src 0.0.0.0/0.0.0.0 http_access allow all cache_mgr root cache_effective_user proxy cache_effective_group proxy httpd_accel_host192.168.1.1 httpd_accel_port 81 logfile_rotate 0 log_icp_queries off buffered_logs on	الوصول للجميع لكل شيء المسؤول عن البريد UID GID خادم الوب الحقيقي المنفذ

بهذه الطريقة، يُعطي الخيار httpd_accel_host إمكانية تشغيله كخادم وسيط. لمزيد من المعلومات زُر

.http://www.squid-cache.org

6.1 سكويد proxy-caching

بهذه الطريقة، يمكن لسكويد التحكم بالوصول إلى الإنترنت، من حيث وقت الوصول والعناصر التي يمكن الوصول إليها.

في هذه الحالة، سيكون على ملف الإعداد أن يحوي التعديلات التالية مضافة إلى /etc/squid.conf:

```
acl localnet src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255
acl Safe_ports port 80 443 210 70 21 102565535
acl CONNECT method CONNECT
acl all src 0.0.0.0/0.0.0.0
http_access allow localnet
http_access allow localhost
http_access deny
http_access deny CONNECT
http_access deny all
cache_emulate_httpd_log on
```

الاختلاف الرئيسي بينه وبين الوضع الآخر هي سطور acl – والتي يسمح فيها للعملاء من المجموعة C الشبكة 192,168,1,0 بالوصول إلى الخادم الوسيط –، وأيضاً العنوان المحلي، والمنافذ الأخرى التي يمكنها الوصول إلى الإنترنت، وهي http 80 و https 443 و whois 210 و gopher 70 و ftp 21، وأيضاً الطريقة connect ممنوعة لتجنب الوصول إلى الخادم الوسيط من الخارج، ومن ثم فإن كل العناوين والمنافذ في الخادم الوسيط ممنوعة. المزيد من المعلومات على http://www.squid-cache.org، وللخادم transparent على http://tldp.org/HOWTO/TransparentProxy-1.html.

يعني LDAP الميفاق الخفيف للوصول للدليل lightweight directory access protocol، وهو ميفاق للوصول إلى البيانات اعتماداً على خدمة X,500. يتم تشغيله على TCP/IP، والمجلد أشبه ما يكون بقاعدة بيانات تحوي معلومات معتمدة على خصائص. يسمح النظام لهذه المعلومات بأن تُنظَّم بطريقة آمنة، وأن تستخدم نُسخاً لتبقى متاحة، والتأكد من اتساقها، والتحقق من البيانات التي تم الوصول إليها أو تعديلها.

تعتمد الخدمة على نموذج العميل-الخادم وهناك واحد أو أكثر من الخوادم التي تحوي البيانات؛ عندما يتصل عميل ويطلب معلومات، يرد الخادم بالبيانات أو بمؤشر إلى خادم آخر يمكن استخراج مزيد من المعلومات منه، لكن يمكن للعميل أن يرى فقط دليلاً بالمعلومات العامّة.

لاستيراد أو تصدير معلومات بين خوادم ldap أو لوصف عدد من التغييرات التي سيتم تطبيقها على الدليل، نستخدم صيغة تدعى LDIF (اختصاراً لعبارة LDAP data interchange format). تختزن LDIF المعلومات بهيكلية موجهة للكائنات، ويتم تحويلها لاحقاً إلى الصيغة الداخلية لقاعدة البيانات. لملف LDIF صيغة تشبه:

```
dn: o = UOC, c = SP
or: UOC
objectclass: organization
dn: cn = Pirulo Nteum, o = UOC, c = SP
cn: Pirulo Nteum
sn: Nteum
mail: nteum@uoc.edu
objectclass: person
```

تحدَّد كل مدخلة باسم يشار إليه بأنه اسم مميز distinguished name – dn. يتكون dn من اسم المدخلة إضافة إلى سلسلة من الأسماء التي تربطه بهيكلية الدليل ومكان وجود objectclass التي تحدد الخصائص التي يمكن أن تستخدم في هذه المدخلة. تقدم LDAP مجموعة أساسية من تصنيفات العناصر: مجموعات (وتتضمن قوائم غير مرتبة من العناصر المنفردة أو مجموعات العناصر)، والأماكن (كالدول ووصف كل منها)، والمؤسسات، والأشخاص. يمكن للمدخلة أن تنتمي لأكثر من

تصنيف، فعلى سبيل المثال يندرج الفرد تحت تصنيف "الأشخاص"، لكن يمكن أيضاً أن يحدد بخصائص التصنيفات inetOrgPerson و groupOfNames و organisation. تحدد هيكلية عناصر الخادم (وتدعى أنماطاً schema) الخصائص المسموح بها لعنصر ضمن تصنيف (وهي محددة في /etc/ldap/schema مثل openldap.schema, cobra.schema, nis.schema, inetorgperson.schema ...إلخ).

كل البيانات معروضة كزوج [الخاصية = القيمة]، حيث "الخاصية" (attribute) وصف للمعلومات التي تحويها، على سبيل المثال، الخاصية المستخدمة لتخزين اسم شخص هي commonName أو cn، وبعبارة أخرى، شخص يدعى Pirulo Nteum سيتم إدراج اسمه كالتالي cn: Pirulo Nteum، وسيكون قد تضمن خصائص أخرى للتصنيف person، مثل givenname: Pirulo و surname: Nteum و mail: pirulo@uoc.edu. للتصنيف خصائص إجبارية وأخرى اختيارية، ولكل خاصية سياق مرتبط بها يوضح نوع البيانات التي تحويها الخاصية، على سبيل المثال، bin (أي binary)، و ces (أي case exact string، يجب أن تتطابق حالة الأحرف في البحث)، و cis (أي case ignore string، يمكن تجاهل حالة الأحرف عند البحث)، tel (أي telephone number string، رقم الهاتف، ويتم فيه تجاهل الفراغات وإشارة "-")، و dn (أي distinguished name، الاسم الفريد). كمثال على ملف بصيغة LDIF:

```
dn: dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
dn: ou = groups, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: groups
dn: ou = people, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: people
dn: cn = Pirulo Nteum, ou = people, dc = UOC, dc = com
cn: Pirulo Nteum
sn: Nteum
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
        uid:pirulo
        userpassword:{crypt}p1pss2ii(0pgbs*do&@ = )eksd
        uidnumber:104
        gidnumber:100
        gecos:Pirulo Nteum
        loginShell:/bin/bash
        homeDirectory: /home/pirulo
```

```
        shadowLastChange:10877
        shadowMin: 0
        shadowMax: 999999
        shadowWarning: 7
        shadowInactive: -1
        shadowExpire: -1
        shadowFlag: 0
dn: cn = unixgroup, ou = groups, dc = UOC, dc = com
objectclass: top
objectclass: posixGroup
cn: unixgroup
gidnumber: 200
memberuid: pirulo other-user
memberuid:
```

يمكن إكمال السطور الطويلة في سطور جديدة تحتها مبدوءة بفراغ أو إزاحة (tab) (في صيغة LDIF). في هذه الحالة، تم تعريف DN الرئيسية للمؤسسة dc=OUC, dc=com التي تحوي وحدتين فرعيتين: people و groups. ومن ثم فقد عرفت شخصاً ينتمي إلى people و group. وبما أننا جهزنا الملف بالبيانات، فعلينا استيراده إلى الخادم بحيث يصير متاحاً لعملاء LDAP. هناك أدوات لتحويل البيانات من قواعد البيانات المختلفة إلى صيغة LDIF.

في دبيان، نحتاج لأن نثبت الحزمة slapd وهي خادم OpenLDAP. أثناء التثبيت، ستسأل مجموعة من الأسئلة، مثل: Method of installing the directory: auto;extensions to the directory [domain-host,site,institution]: host, domain, password of the Adm; replicate local changes to other servers: no. سُنشئ هذا الإعداد ملفاً في /etc/ldap/slapd.conf وقاعدة البيانات في /var/lib/ldap. هناك أيضاً ملف آخر /etc/ldap/ldap.conf (أو ~/.ldaprc) وهو ملف الإعداد المستخدم لإنشاء القيم المبدئية عند تشغيل عملاء ldap. وهي تشير هنا إلى قاعدة البيانات، وخادم ldap، والمعاملات الأمنية، وحجم البحث، إلخ.

ملف إعداد الخادم /etc/ldap/slapd.conf (انظر إلى man slap.conf) وتتكون من أجزاء مختلفة، ويشار إلى كل منها بواحد من التوجيهات التالية: عامة global، وخاصة بالخلفية backend specific، وخاصة بقاعدة البيانات database specific على الترتيب. التوجيه العام له طبيعة عامة، وينطبق على كل النهايات الخلفية (قواعد البيانات)، ويحدد أسئلة عامة

مثل صلاحيات الوصول، والخصائص، ووقت الانتظار، والأنماط schemas، إلخ. يحدد التوجيه الخاص بالخلفية خصائص الخلفية المحددة التي تعرّفها، والتي تحدد (bdb, dnssrv, ldbm، ...)، والتوجيه الخاص بقاعدة البيانات يحدد الخصائص لقاعدة البيانات التي تعرفها. لتشغيل الخادم علينا أن ننفذ:

/etc/init.d/slapd start   (أو stop لإيقافه)

أثناء التثبيت، سيكون النظام قد أنشأ الروابط الصحيحة لتشغيله بعد الإقلاع.

7.1 إنشاء والحفاظ على قاعدة البيانات

هناك طريقتان لإدخال البيانات في قاعدة بيانات LDAP. الأولى سهلة ومناسبة للكميات الصغيرة من البيانات، وهي تفاعلية، و نحتاج لاستخدام أدوات مثل ldapadd (أو أي أداة أخرى مثل LDAP Browser كما في http://www.iit.edu/~gawojar/ldap/) لعمل مدخلات جديدة. الثانية نحتاج للعمل عليها أثناء توقف الخادم، وهي مناسبة لقواعد البيانات الضخمة، وهي تستخدم الأمر slapadd المضمنة في slapd. لأنها عامة أكثر، سنشرح باختصار الطريقة الثانية، والتي يجب فيها أولاً أن نتحقق من أن slapd.conf يحوي الخصائص التالية: suffix (رأس الدليل، على سبيل المثال "o=UOC, c=SP")؛ الدليل /var/lib/ldap/ (المجلد الذي ستُنشأ فيه الفهارس والتي يمكن أن تكتب slapd). يمكننا أيضاً أن نتحقق بأن قاعدة البيانات تحوي تعريف الأدلة التي نرغب باستخدامها:

```
index cn,sn,uid
index objectClass pres,eq
```

بعد تعريفنا لـ slapd.conf، علينا أن ننفذ الأمر:

```
slapadd -l entry-f configuration [-d level] [-n whole| -b suffix]
```

المعاملات هي:

-l ملف بصيغة LDFI.

-f ملف إعداد الخادم الذي يشار فيه إلى كيفية إنشاء الفهارس.

d- مستوى التصحيح.

n- عدد قواعد البيانات، إذا كان لدينا أكثر من واحدة.

b- يحدد أي قاعدة بيانات تحتاج للتعديل.

هناك أوامر أخرى في slapd، مثل slapindex التي تسمح بإنشاء الفهارس، و slapcat التي تسمح بحفظ قاعدة البيانات في ملف بصيغة LDIF.

8 خدمات الملفات NFS

> يسمح NFS للخادم بتصدير نظام ملفات بحيث يمكن استخدامه تفاعلياً من طرف عميل. تتكون الخدمة من الخادم nfsd، وعميل (mountd) يمكن أن يشارك نظام ملفات (أو جزءاً منه) عبر الشبكة.

في دبيان، ثبّت apt-get install nfs-common portmap للعميل، بينما يحتاج الخادم:

```
apt-get install nfs-kernel-server nfs-common portmap
```

الخادم (في دبيان) يبدأ عبر nfscommon و نصوص nfs-kernel-server في /etc/init.d/ (والروابط المناسبة في /etc/rcX.d/).

يستخدم الخادم الملف (/etc/exports) لإدارة الوصول والتحكم في نظام الملفات الذي سيتم الوصول إليه عن بعد. في العميل، الجذر (أو مستخدم آخر عبر sudo)، يمكنه أن يوصل النظام البعيد باستخدام الأمر

```
mount IPserver:remote-directory local_directory
```

ومن تلك اللحظة، سنتمكن من رؤية المجلد البعيد remote-directory في المجلد المحلي local_directory (الذي يجب أن يكون موجوداً قبل تنفيذ الأمر mount). هذه العملية في العميل يمكن أن تتم أتمتتها باستخدام ملف الضم الآليّ (/etc/fstab) بإضافة سطر كالتالي مثلاً:

```
pirulo.remix.com:/usr/local /pub nfs rsize=8192,wzise=8192,timeo=14
```

تشير هذه الجملة إلى أن المجلد /usr/local/ في المضيف pirulo.remix.com سيتم ضمّه في المجلد المحلي /pub/. المعاملان rsize و wsize هما حجم كل القراءة والكتابة، و timeo هو RPC timeout (إذا كانت هذه المعاملات الثلاثة غير محددة، فسيتم أخذ القيم المبدئية).

يخدم الملف /etc/exports كقائمة تحكم بالوصول access control list – ACL لنظام الملفات الذي يمكن تصديره إلى العملاء. يحوي كل سطر نظام ملفات يتم تصديره متبوعاً بالعميل الذي يمكنه ضمّه، مفصولة بسطور فارغة. يمكن لكل عميل أن يحصل على مجموعة من الخيارات المرتبطة به من أجل تعديل سلوكه (انظر إلى man exports لقائمة مفصلة بالخيارات).

كمثال على ذلك:

```
# Example of /etc/exports
/ /master(rw) trusty(rw,no_root_squash)
/projects proj*.local.domain(rw)
/usr *.local.domain(ro) @trusted(rw)
/pub (ro,insecure,all_squash)
/home 195.12.32.2(rw,no_root_squash) www.first.com(ro)
/user 195.12.32.2/24(ro,insecure)
```

السطر الأول يصدر نظام الملفات بأكمله (/) لكل من master و trusty في الوضع قراءة/كتابة. إضافة إلى ذلك، فلا يوجد uid squashing (سيصل الجذر في العميل كجذر إلى ملفات الجذر للخادم، وبعبارة أخرى، الجذران متكافئان رغم كونهما من جهازين مختلفين، وهذه مناسبة للأجهزة عديمة الأقراص). السطران الثاني والثالث يظهران أمثلة على "*" و netgroups (يشار إليها بالرمز @). يصدر السطر الرابع المجلد /pub/ لأي جهاز في العالم، للقراءة فقط، ويسمح بالوصول لعملاء NFS الذين يستخدمون منفذاً محجوزاً لخدمة NFS (الخيار insecure)، وكل شيء يتم تنفيذه ضمن المستخدم nobody (الخيار all squash). السطر الخامس يحدد عميلاً واحداً لعنوانه، والثاني نفسه لكن مع قناع شبكة (/24) ومع خيارات بين قوسين () ودون أيّة فراغات. يمكن أن يكون هناك فراغات بين العملاء المفعلين فقط. علينا أن نبقي في بالنا أن هناك 3 إصدارات لـ NFS، وهي (v2 و v3 و v4). الأكثر شيوعاً هي v3. إذا كنا نتصل من عميل v3 إلى خادم v2، فيجب الإشارة إلى هذا بعامِل.

9 خادم ويكي Wiki

الويكي (من لغة هاواي wiki wiki وتعني "سريع" أو "بسرعة") هو خادم تعاوني يمكن أن يحرر فيه العديد من المستخدمين الذين يمكنهم إضافة وتحرير وحذف وتعديل محتوى صفحة وب بطريقة سهلة وسريعة وتفاعلية. تسمح تقنية الويكي لصفحات الوب المخزنة في خادم عام (صفحات ويكي) التي ستكتب بشكل تعاوني عبر متصفح باستخدام إشارة بسيطة لتنسيق معين، وإنشاء روابط، إلخ، بحفظ سجل بالتغييرات التي تجعل من الممكن استرجاع أي حالة سابقة لصفحة بسهولة. عندما يعدل أحد ما صفحة ويكي، تظهر تغييراتها مباشرة على الوب، دون المرور على أي نوع من المراجعة المسبقة. يمكن أن يشير الويكي أيضاً إلى صفحات النصوص المتشعبة، التي يمكن زيارتها وتعديلها من أي شخص (تعريف ويكيبيديا). لدبيان الويكي الخاص بها في http://wiki.debian.org وفيدورا في http://fedoraproject.org/wiki/، وكلاهما مبنيان على Moin Moin (الموقع الرسمي http://moinmoin.wikiwikiweb.de). إن MoinMoin هو WikiClone مكتوب بلغة بايثون يمكنه إنشاء ويكي بسرعة، يحتاج فقط خادم وب ولغة بايثون المثبتة.

في http://moinmoin.wikiwikiweb.de/MoinMoinPackages/DebianLinux يمكننا أن نجد تعليمات تفصيلية لتثبيت MoinMoin في دبيان، لكن - بشكل أساسي - تندرج تحت العناوين التالية: 1) تثبيت أباتشي 2، 2) ضبط أباتشي ليلاحظ Moin Moin، 3) تثبيت حزمة MoinMoin، 4) ضبط MoinMoin، 5)إعادة تشغيل أباتشي. مثال على الإعداد:

```
apt-get install python-moinmoin
mkdir /var/www/mywiki
cp -r /usr/share/moin/data /usr/share/moin/underlay \
/usr/share/moin/server/moin.cgi /var/www/mywiki
chown -R www-data:www-data /var/www/mywiki
chmod -R g+w /var/www/mywiki
```

- ضبط أباتشي 2 بإضافة /etc/apache2/conf.d/wiki (أو أي مسار يدل على مكان ملف الإعداد):

```
Alias /wiki/ "/usr/share/moin/htdocs/"
<Location /mywiki>
	SetHandler python-program
	PythonPath "['/var/www/mywiki','/etc/moin/']+sys.path"
	PythonHandler MoinMoin.request::RequestModPy.run
	PythonDebug On
</Location>
```

- إعادة تشغيل أباتشي 2:

```
/etc/init.d/apache2 reload
```

- ضبط MoinMoin: عدل /etc/moin/farmconfig.py (العديد من الويكي)

```
wikis =</emphasis>
[
("mywiki", r"^yoursite.com/mywiki/.*$"),
]
```

- يمكننا أن نستخدم أيضاً (ويكي واحدة فقط):

```
wikis = [
("mywiki",r".*"),
]
```

- وأيضاً في /etc/moin/farmconfig.py أزل التعليق عن data_dir و data_underlay_dir (واحد لكل ويكي)، وانسخ الملف.

```
cp /etc/moin/moinmaster.py /etc/moin/mywiki.py
```

- ثم عدّل /etc/moin/mywiki.py وغيّر:

```
sitename = u'MyWiki'
data_dir = '/var/www/mywiki/data'
data_underlay_dir = '/var/www/mywiki/underlay'
```

سيتم تثبيت الويكي في http://yoursite.com/mywiki/.

الأنشطة

١) اضبط خادم DNS كخادم اختزان مؤقت وبنطاقه الخاص.

٢) اضبط عميل/خادم NIS بجهازين بتصدير مجلدات مستخدمي الخادم عبر NFS.

٣) اضبط خادم SSH للوصول من جهاز آخر دون كلمة مرور.

٤) اضبط خادم أباتشي + PHPMyAdmin + MySQL + PHP + SSL من أجل جعل الصفحات الشخصية للمستخدمين مرئيّة.

٥) أنشئ واضبط نظام بريد إلكتروني عبر Exim, Fetchmail, Spam-Assassin، وخادم IMAP لاستقبال بريد من الخارج، ولتكون قادراً على قراءتها من جهاز بعيد بعميل موزيلا (Thunderbird).

٦) ثبت الويكي Moin Moin، وأنشئ مجموعة من الصفحات للتأكد من أنها تعمل.

المراجع

مصادر أخرى للمراجع والمعلومات:

[Debc, LPD03b, Ibi]

http://tldp.org/HOWTO/DNS-HOWTO-7.html

http://tldp.org/HOWTO/NIS-HOWTO/verification.html

Squid proxy server

Proxy Cache: http://www.squid-cache.org/

Transparent Proxy: http://tldp.org/HOWTO/TransparentProxy-1.html

Proftpd: http://www.debian-administration.org/articles/228

PureFtpd: http://www.debian-administration.org/articles/383

Exim: http://www.exim.org/docs.html

Mutt: http://www.mutt.org

ProcMail: http://www.debian-administration.org/articles/242

LWP:
http://www.lawebdelprogramador.com/temas/tema_stablephpapachemysql.php

Moin Moin: http://moinmoin.wikiwikiweb.de/

Moin Moin + Debian:
http://moinmoin.wikiwikiweb.de/MoinMoinPackages/DebianLinux

Apache2 + SSL: http://www.debian-administration.org/articles/349

إدارة البيانات

د. رِمو سُـپي بُلدريتو

مقدمة

من النواحي المهمة لنظام التشغيل مكان وكيفية تخزين البيانات. عندما نحتاج لأن تكون البيانات متاحة بكفاءة، فمن الضروري استخدام قواعد البيانات databases – DB.

قاعدة البيانات مجموعة منظمة من البيانات التي يمكن أن تُنَظَّم بطريقة بسيطة وفعالة بواسطة مدير قواعد البيانات. قواعد البيانات الحالية معروفة بأنها علاقيّة[1]، حيث يمكن تخزين هذه البيانات في جداول مختلفة لتسهيل الإدارة. لهذا الغرض، ومن أجل جعل الوصول إلى قواعد البيانات يتم بطريقة معيارياً (متفق عليها) يتم استخدام لغة تعرف بالاسم SQL (اختصاراً لعبارة structured query language). تسمح هذه اللغة بالتفاعل بسرعة ومرونة بغض النظر عن برنامج قواعد البيانات.

في الوقت الحاضر، تتكون أكثر الطرق المستخدمة شيوعاً من الوصول إلى قاعدة بيانات من تطبيق يشغّل أكواد SQL. على سبيل المثال، من الشائع جدًّا الوصول إلى قاعدة بيانات عبر صفحة وب تحوي أكواد PHP أو Perl (الأكثر شيوعاً). عندما يطلب عميلٌ صفحةً ما، يتم تنفيذ كود PHP أو Perl المضمّن في الصفحة التي يتم تشغيلها[2]، يتم الوصول إلى قاعدة البيانات وإنشاء الصفحة بمحتواها الثابت والمحتوى الذي تم استخراجه من قاعدة البيانات والتي يتم إرسالها لاحقاً إلى العميل. من أفضل الأمثلة على قواعد البيانات الحالية تلك التي تقدمها PostgreSQL و MySQL وهما اللذان سنبحثهما.

ولكن عندما نعمل على تطوير التطبيقات، فهناك نواحٍ أخرى ذات علاقة يجب أخذها بعين الاعتبار، وتتعلق بجودتها وبيئتها (خاصة إذا كانت هناك مجموعة من المستخدمين الذين يعملون على نفس البيانات). هناك العديد من حزم التحكم بالإصدارات (المراجعات)، لكن الغرض منها كلها تسهيل إدارة الإصدارات المختلفة لكل منتج يتم تطويره إضافة إلى التخصيصات المحتملة المعمولة لأيّ عميل بعينه.

---

1 نسبة إلى كلمة "علاقة"، حيث تحوي وتعتمد في عملها على علاقات بين العناصر (الجداول) المختلفة.

2 إن ما يتم حقيقة هو أن كود PHP أو Perl يتم تنفيذه جهة الخادم قبل و/أو بعد إنشاء الصفحة، أو عند تنفيذ طلب ما، وكثيراً ما يتم إنشاء الصفحة المعروضة أو جزءاً منها بواسطة كود PHP أو Perl أو لغة برمجة تنفَّذ جهة الخادم. للمزيد حول الموضوع، ابحث في الوب عن server-side programming و client-side programming.

تُقدَّم أنظمة التحكم بالإصدارات للتحكم بالإصدارات المختلفة للمصدر البرمجي. لكن المفاهيم نفسها تنطبق على المناطق الأخرى، ليس فقط المصدر البرمجي، بل والمستندات، والصور، إلخ أيضاً. ورغم أنه يمكن أيضاً تنفيذ نظام تحكم بالإصدارات يدوياً، إلا أنه يُنصح وبشدّة بالحصول على أدوات تسهّل هذه المهمّة[3] (cvs, subversion, SourceSafe, ClearCase, Darcs, Plastic SCM, RCS، إلخ.).

سنشرح في هذا الجزء أداة Subversion للتحكم وإدارة مراجعات ملفات عديدة، وأتمتة التخزين، والقراءة، والتعرف على ودمج المراجعات المختلفة. هذه البرامج مفيدة عندما تكون هناك نصوص يتم مراجعتها بكثرة وتتضمن مصادر برمجية، وبرامج تنفيذيّة، ومكتبات، وتوثيق، ورسوميات، ومقالات، وملفات أخرى.

السبب وراء استخدام cvs و Subversion هو أن cvs قد كان من أكثر هذه الحزم شيوعاً يوماً ما، وأن Subversion أحد البدائل الحالية الشائعة لـ cvs والتي حلّت العديد من مشاكلها. تعرف Subversion أيضاً بالاسم svn، حيث أن هذا اسم أداة سطر الأوامر. من المزايا الهامة في Subversion – غير الموجودة في CVS – هي أن الملفات ومراجعاتها لا يوجد لكلّ منها رقم مراجعة مستقل. بدلاً من هذا، هناك للمستودع بأكمله رقم إصدار وحيد يحدد حالة مشتركة لكل ملفات المستودع في نقطة زمنية محددة.

---

3 أكثر هذه الأدوات شيوعاً هي svn و git، وهناك أدوات لم تعد شائعة بسبب وجود بدائل أفضل منها مثل cvs.

PostgreSQL 1

تستخدم لغة قاعدة بيانات PostgreSQL نموذج عميل وخادم. تتكون جلسة PostgreSQL من مجموعة برامج تتعاون فيما بينها:

- عملية الخادم التي تتعامل مع ملفات قاعدة البيانات تقبل اتصالات من عملاء وتقوم بالمهام التي يطلبها العملاء من قاعدة البيانات. برنامج الخادم في PostgreSQL يُدعى postmaster.
- تطبيق العميل (الواجهة الأمامية) هي التي تطلب أن تتم العمليات على قاعدة البيانات، ويمكن أن تتنوع هذه العمليات بشكل كبير، على سبيل المثال: أدوات في الوضع النصّيّ، والرسومي، وخوادم الوب، إلخ.

غالباً يكون العميل والخادم على مضيفين مختلفين ويتواصلان عبر اتصال TCP/IP. يمكن أن يقبل الخادم عدّة اتصالات من عملاء مختلفين، وتشغيل عملية تستمع إلى طلبات مستخدم بعينه، وتنفيذ واحدة جديدة تلقائيًّا لكل اتصال جديد. هناك مجموعة من المهام التي يمكن أن ينفّذها المستخدم أو المدير - حسبما يكون مناسباً - والتي سنشرحها فيما يلي.

1.1 كيف يمكننا إنشاء قاعدة بيانات

الخطوة الأولى لفحص ما إذا كان من الممكن الوصول إلى خادم قواعد البيانات هي إنشاء قاعدة بيانات. يمكن لخادم PostgreSQL أن يتعامل مع عدد كبير من قواعد البيانات ويُنصح باستخدام واحدة مختلفة لكل مشروع. لإنشاء قاعدة بيانات، نستخدم الأمر createdb من سطر أوامر نظام التشغيل. سيُنشئ هذا الأمر رسالة CREATE DATABASE إذا كان كل شيء صحيحاً. من المهم أن نأخذ بعين الاعتبار أنه للقيام بهذا الأمر يجب أن يكون لدينا مستخدم يُسمح له بإنشاء قاعدة بيانات. في القسم المتعلق بالتثبيت (1،4)، سنرى أن هناك مستخدماً - المستخدم الذي يثبّت قاعدة البيانات - لديه صلاحيات لإنشاء قواعد بيانات، وإنشاء مستخدمين جدد يمكنهم إنشاء قواعد بيانات. وبشكل عام (في دبيان) فالمستخدم المبدئي هو postgres. ولهذا، قبل تشغيل createdb، سنحتاج لتشغيل postgres (إذا كنا في طور المستخدم الجذر، فلن نحتاج لكلمة مرور، ولكن أي مستخدم آخر سيحتاج لكلمة مرور المستخدم postgres)، ومن ثم سنكون قادرين على تنفيذ

createdb. لإنشاء قاعدة بيانات اسمها nteumdb:

```
createdb nteumdb
```

إذا لم يكن بإمكاننا إيجاد الأمر[4]، فقد يكون السبب هو كون المسار غير مضبوط بشكل صحيح، أو أن قاعدة البيانات غير مثبتة بشكل صحيح. يمكننا أن نجرب بالمسار الكامل (/usr/local/pgsql/bin/createdb nteumdb)، والذي سيعتمد على التثبيت لدينا، أو تفقّد المراجع لحل المشكلات. هناك رسائل أخرى قد تظهر، مثل could not connect to server (أي "تعذّر الاتصال بالخادم") عندما لا يكون الخادم يعمل، أو CREATE DATABASE: permission denied عندما لا تكون لدينا صلاحيات لإنشاء قاعدة البيانات. لحذف قاعدة البيانات، يمكننا أن نستخدم dropdb nteumdb.

1.2 كيف يمكننا الوصول إلى قاعدة بيانات

بعد أن نكون قد أنشأنا قاعدة بيانات، يمكننا الوصول إليها بطرق عديدة:

- بتنفيذ أمر تفاعليّ يُدعى psql، الذي يسمح لنا بتحرير وتنفيذ أوامر SQL.
- تنفيذ واجهة رسومية، مثل PgAccess، أو حزمة بها دعم ODBC لإنشاء التعامل مع قواعد البيانات.
- كتابة تطبيق باستخدام إحدى اللغات المدعومة، مثل PHP, Perl, Java، ... (انظر إلى PostgreSQL 7,3 Programmer's Guide).

أسباب تتعلق بالبساطة، يمكننا أن نستخدم psql للوصول إلى قاعدة بيانات، مما يعني أنه سيكون علينا الدخول إلى psql nteumdb: ستظهر بعض الرسائل المتعلقة بالإصدار ومعلومات ومحث يشبه <= nteumdb. يمكننا أن ننفّذ بعض أوامر SQL التالية:

```
SELECT  version();        أو أيضاً        SELECT  current date;
```

في psql أيضاً أوامر ليست ضمن SQL وتبدأ بالشرطة العكسية "\"، مثل \h (قائمة بكل الأوامر المتاحة) أو \q للإنهاء.

---

4 أي إذا عثرنا على رسالة خطأ command not found عند تنفيذ الأمر.

مثال

الوصول إلى قاعدة البيانات nteumdb :

psql nteumdb (ثم زر الإدخال)

nteumdb =>

1.3 لغة SQL

ليس الهدف من هذا القسم تقديم درس تعليمي عن SQL، ولكننا سنحلل بعض الأمثلة لنرى إمكانات هذه اللغة. هذه أمثلة تأتي مع توزيعة PostgreSQL في المجلد InstallationDirectory/src/tutorial (حيث InstallationDirectory هو مجلد تثبيت PostgreSQL)؛ من أجل الوصول إليها، انتقل إلى مجلد PostgreSQL (بالأمر cd InstallationDirectory/src/tutorial) ثمّ نفّذ psql -s nteumdb، وما إن تدخل نفّذ i basics.sql\. المعامل i\ يقرأ الأوامر في الملف المحدّد (وهو في هذه الحالة basics.sql).

إن PostgreSQL نظام إدارة قواعد بيانات علاقية RDBMS (أي Relational Database Management System)، مما يعني أنه يدير بيانات مخزنة في جداول. لكل جدول عدد معين من الصفوف والأعمدة، ويحوي كل عمود نوعاً معيناً من البيانات. الجداول مجموعة في قاعدة بيانات واحدة، وهناك خادم واحد يحوي مجموعة من قواعد البيانات (تدعى هذه المجموعة الكاملة تجمّعاً من قواعد البيانات database cluster).

لإنشاء جدول في psql مثلاً، نفّذ:

```
CREATE TABLE weather (
        city            varchar(80),
        min_temp        int,
        max_temp        int,
        real            rain,
        day             date
);
```

مثال

إنشاء جدول. في psql نفذ:

CREATE TABLE NameTB (var1 type, var2 type, …);

ينتهي الأمر عندما نكتب الفاصلة المنقوطة ";" ويمكننا أن نستخدم الفراغات والإزاحات بحرية. يحدد varchar(80)

هيكلية البيانات التي يمكن أن تختزن حتى 80 محرفاً (في هذه الحالة). النقطة المذكورة لنوع معين من PostgreSQL.

لحذف الجدول:

```
DROP TABLE table_name
```

يمكننا إدخال البيانات بطريقتين، الأولى هي إدخال كل بيانات الجدول، والثانية هي تحديد المتغيرات والقيم التي نرغب بتعديلها.

```
INSERT INTO weather VALUES ('Barcelona', 16, 37, 0،25, '2007-03-19');
INSERT INTO weather (city, min_temp, max_temp, rain, day)
```

الطريقة المستخدمة في السطر الأول من السطرين السابقين يمكن أن يكون بسيطاً للكميات القليلة من البيانات، ولكن عندما يكون علينا إدخال كم كبير من البيانات، فيمكن أن ننسخها من ملف بالجملة:

```
COPY weather FROM '/home/user/weather.txt';
```
(يجب أن يكون هذا الملف موجوداً على الخادم وليس العميل)

لنطّلع على جدول ما، يمكننا أن ندخل:

```
SELECT * FROM weather;
```

حيث تعني إشارة النجمة كل الأعمدة.

أمثلة

- إدخال البيانات إلى الجدول. في psql:

```
INSERT INTO TBName (valueVar1, valueVar2, …);
```

- استيراد البيانات من ملف. في psql:

```
COPY TBName FROM 'FileName';
```

- عرض البيانات. في psql:

```
Select * FROM TBName;
```

من الأمثلة على الأوامر الأكثر تعقيداً (في psql) :

- عرض العمود city بعد كتابة:

```
SELECT city, (max_temp+min_temp)/2 AS average_temp, date FROM weather;
```

- عرض كل شيء عند تحقق الجملة الشرطية:

```
SELECT * FROM weather WHERE city = 'Barcelona'
        AND rain \verb+>+ 0،0;
```

- دمج الجداول:

```
SELECT * FROM weather, city WHERE city = name;
```

- الدوال functions، وفي هذه الحالة الدالة max (الأكبر):

```
SELECT max(min_temp) FROM weather;
```

- الدوال المتداخلة:

```
SELECT city FROM weather WHERE min_temp = (SELECT max(min_temp) FROM
weather);
```

- التعديل الانتقائيّ:

```
UPDATE weather SET max_temp = max_temp 2, min_temp = min_temp 2
        WHERE day > '19990128';
```

- حذف السجلّ:

```
DELETE FROM weather WHERE city = 'Sabadell';
```

1.4 تثبيت PostgreSQL

هذه الخطوة هامة لمدراء قواعد البيانات. تتضمن مهام مدير قواعد البيانات تثبيت البرمجيات، وتشغيلها، وإعدادها، وإدارة المستخدمين، ومهام قواعد البيانات وصيانتها.

يمكن تثبيت قاعدة البيانات بطريقتين: عبر الحزم الثنائية (التنفيذية) للتوزيعة – وهذه ليست بالمهمة الصعبة، حيث أن النصوص البرمجية للتوزيعة تقوم بكل الخطوات الضرورية لجعل قاعدة البيانات تعمل – أو عبر المصدر البرمجي – الذي يتوجب علينا تصريفه وتثبيته –. في الحالة الأولى، يمكننا استخدام kpackage أو apt-get – في دبيان. الحالة الثانية – ننصح دائمًا بالذهاب إلى المصدر (أو إلى مستودع مرآة للنسخة الأصلية). من المهم أن نبقي في بالنا أن التثبيت من المصدر سيبقى خارج قاعدة بيانات البرمجيات المثبتة وأن فوائد إدارة البرمجيات المقدمة – كتلك التي يقدمها كل من apt-cache و apt-get – لن تكون متوفرة.

التثبيت من المصدر البرمجي خطوة بخطوة:

- نحتاج أولاً للحصول على المصدر البرمجي من الموقع http://www.postgresql.org/download وفك ضغطه (حيث x.x.x هو رقم الإصدار المتاح، وهو 8,2,3 أثناء تأليف الكتاب):

```
gnuzip postgresql-x.x.x.tar.gz
tar -xf postgresql-7،3.tar
```

- انتقل إلى مجلد postgresql واضبطه بالأمر configure/.
- صرّفه باستخدام gmake، وتحقق من التصريف بالأمر gmake check، ثم ثبّته بالأمر gmake install (في الوضع المبدئيّ، سيتم التثبيت في usr/local/pgsql/).

1.4.1 ما بعد التثبيت

فعّل المتغيرات في الصدفة bash, sh, ksh:

```
LD_LIBRARY_PATH = /usr/local/pgsql/lib;
PATH = /usr/local/pgsql/bin:$PATH;
export LD_LIBRARY_PATH PATH;
```

أو في حالة csh:

```
setenv LD_LIBRARY_PATH /usr/local/pgsql/lib;
set path = (/usr/local/pgsql/bin $path)
```

ننصح بوضع هذا الإعداد في النص البرمجي لإعداد المستخدم، على سبيل المثال etc/profile/ أو bashrc. في حالة bash. للحصول على وصول إلى أدلة الاستخدام، فسيكون علينا ضبط متغير MANPATH بنفس الطريقة:

```
MANPATH = /usr/local/pgsql/man:$MANPATH;
export MANPATH
```

بمجرد تثبيت قاعدة البيانات، سنحتاج لإنشاء مستخدم يتحكم بقواعد البيانات (يُنصح بإنشاء مستخدم يختلف عن المستخدم الجذر، بحيث لا يكون هناك صلة بينها وبين الخدمات الأخرى في الجهاز، كالمستخدم postgres باستخدام الأمر useradd مثلاً.

ومن ثم سنحتاج لإنشاء منطقة تخزين لقواعد البيانات (مكان واحد) على القرص، والذي سيكون مجلداً ، مثل usr/local/pgsql/data/. ولأجل هذا، نفّذ initdb -D /usr/local/pgsql/data بالمستخدم الذي أنشأته في الخطوة

السابقة. قد نتلقى رسالة تفيد بأنه تعذّر إنشاء المجلد بسبب عدم وجود صلاحيات، مما يعني أنه سيكون علينا أولاً إنشاء المجلد ومن ثم إخبار قاعدة البيانات بمكانه، بصلاحيات المستخدم الجذر علينا أن ننفذ التالي:

```
mkdir /usr/local/pgsql/data
chown postgres /usr/local/pgsql/data
su postgres
initdb -D /usr/local/pgsql/data
```

شغّل الخادم (المسمى postmaster)، ولعمل ذلك استخدم:

```
postmaster -D /usr/local/pgsql/data
```

لتشغيله في الوضع الظاهر (في الواجهة)، أما لتشغيله في الوضع الخفيّ (في الخلفية) استخدم:

```
postmaster -D /usr/local/pgsql/data < logfile 2 >&1 &
```

يتم عمل إعادات التوجيه من أجل تخزين أخطاء الخادم. تتضمن الحزمة أيضاً نصًّا برمجياً (pg_ctl) بحيث لا نضطر لحفظ كل سياق postmaster من أجل تشغيله:

```
/usr/local/pgsql/bin/pg_ctl start -l logfile \
-D /usr/local/pgsql/data
```

يمكننا إيقاف تشغيل الخادم بطرق مختلفة، كاستخدام pg_ctl، أو مباشرة باستخدام الأمر:

```
kill -INT 'head -l /usr/local/pgsql/data/postmaster.pid'
```

1.4.2 مستخدمو قاعدة البيانات

إن مستخدمي قاعدة البيانات مختلفون عن مستخدمي نظام التشغيل. في بعض الحالات، قد يكون من المفيد لهم أن يبقوا على توافقية بينهما[5]، لكن هذا ليس ضرورياً. المستخدمون لكل قواعد البيانات التي يتحكم بها الخادم، وليس لكل قاعدة بيانات على حدة. لإنشاء مستخدم، نفذ جملة SQL التالية:

---

5 يقصد بهذا أن يكون حسابا المستخدم في النظام وفي قاعدة البيانات بنفس الاسم، بحيث يصير من الأسهل التعرف على المستخدم وتذكر اسمه، أو على الأقل أن يكون هناك تشابه في أجزاء معينة من الاسم، لكن هذا ليس ضروريًّا.

```
CREATE USER name
```

لحذف مستخدمين:

```
DROP USER name
```

يمكننا أيضاً أن نعتمد على البرنامجين createuser و dropuser من سطر الأوامر. هناك مستخدم مبدئيّ يُدعى postgres (في قاعدة البيانات)، مما يسمح لنا بإنشاء البقية (لإنشاء مستخدم جديد ندخل أولاً بالأمر psql -U postgres إذا لم يكن مستخدم نظام التشغيل المستخدَم لإدارة قاعدة البيانات postgres).

يمكن أن يكون لمستخدم قاعدة البيانات مجموعة من المعاملات اعتماداً على ما هو مسموح للمستخدم بفعله:

- superuser: ليس لهذا المستخدم قيود. على سبيل المثال، يمكنه إنشاء مستخدمين آخرين، لعمل هذا نفّذ:

```
CREATE USER name CREATEUSER
```

- منشئ قاعدة البيانات: مصرح له بإنشاء قاعدة بيانات. لإنشاء مستخدم بهذه الخصائص استخدم الأمر:

```
CREATE USER name CREATEDB
```

- كلمة المرور: ضرورية فقط إذا كنا نرغب بالتحكم بوصول المستخدمين عندما يتصلون بقاعدة بيانات لأسباب أمنية. لإنشاء مستخدم بكلمة مرور، يمكننا أن نستخدم:

```
CREATE USER name PASSWORD 'password'
```

- يمكننا أن نغير خصائص مستخدم باستخدام الأمر ALTER USER. يمكننا أيضاً أن ننشئ مجموعات مستخدمين تتشارك نفس الصلاحيات بالأمر:

```
CREATE GROUP GroupName
```

ولإدخال المستخدم في هذه المجموعة:

```
ALTER GROUP GroupName ADD USER Name1
```

أو لحذفه

```
ALTER GROUP GroupName DROP USER Name1
```

مثال:

عمليات المجموعات في psql

```
CREATE GROUP GroupName;
ALTER GROUP GroupName ADD USER Name1 …;
ALTER GROUP GroupName Drop USER Name1 …;
```

عندما ننشئ قاعدة بيانات، فالصلاحيات للمستخدم الذي يُنشئها (وللمستخدم superuser). للسماح لمستخدم آخر باستخدام قاعدة البيانات هذه أو جزءاً منها، فعلينا أن نعطيه صلاحيات. هناك أنواع مختلفة للصلاحيات، مثل SELECT, INSERT, UPDATE, DELETE, RULE, REFERENCES, TRIGGER, CREATE, TEMPORARY, EXECUTE, USAGE, ALL PRIVILEGES (تفقد المراجع لمعرفة معنى كلٍّ منها). لإعطاء صلاحيات، يمكننا أن نستخدم:

```
GRANT UPDATE ON object TO user;
```

حيث يجب أن يكون user مستخدم PostgreSQL صحيحاً، وأن يكون object عنصراً (جدولاً مثلاً). يجب أن يتم تنفيذ هذا الأمر من طرف superuser أو مالك قاعدة البيانات. المستخدم PUBLIC يمكن أن يستخدَم كمرادف لـ"كل المستخدمين"، و ALL كمرادف لـ"كل الصلاحيات". على سبيل المثال، لسحب كل الصلاحيات من كل مستخدمي عنصر ما، يمكننا تنفيذ التالي:

```
REVOKE ALL ON object FROM PUBLIC;
```

1.5 الصيانة

إن مدير قواعد البيانات مسؤول عن مجموعة من المهام التي يجب القيام بها دورياً:

١) **استعادة المساحة:** علينا أن ننفذ الأمر VACCUM دورياً، والذي سيقوم بدوره باستعادة مساحة القرص للصفوف المحذوفة أو المعدلة، وتحديث الإحصائيّات التي يستخدمها برنامج جدولة PostgreSQL، وتحسين ظروف الوصول.

٢) إعادة الفهرسة: في بعض الحالات، يمكن أن تعطي PostgreSQL أخطاء عند إعادة استخدام الفهارس، ولهذا يُنصَح باستخدام الأمر REINDEX دورياً للتخلص من الصفحات والصفوف. يمكننا أيضاً أن نستخدم contrib/reindexdb من أجل إعادة فهرسة قاعدة البيانات بأكملها (علينا أن نأخذ بعين الاعتبار أن هذه الأوامر

تستغرق بعض الوقت اعتماداً على حجم قواعد البيانات).

٣) غير ملفات التقارير: نحتاج لأن نمنع ملفات التقارير من أن تصبح ضخمة جدًّا ويصعب التعامل معها. يمكن عمل هذا بسهولة عندما يكون الخادم يعمل بالأمر:

```
pg_ctl start | logrotate
```

يقوم logrotate بإعادة تسمية ملفات التقارير وإنشاء ملفات جديدة، ويمكن ضبطه بتعديل etc/logrotate.conf/.

٤) النسخ الاحتياطي والاستعادة: هناك طريقتان لحفظ البيانات، بجملة SQL Dump أو بحفظ ملف قاعدة البيانات. الأول سيكون كالتالي:

```
pg_dump DBFile > BackupFile
```

للاستعادة، يمكننا استخدام: psql DBFile < BackupFile

من أجل حفظ كل قواعد بيانات الخادم، يمكننا تنفيذ:

```
pg_dumpall > TotalBackupFile
```

الطريقة الأخرى هي أن تحفظ ملفات قواعد البيانات على مستوى نظام التشغيل، باستخدام هذا الأمر مثلاً:

```
tar -cf backup.tar /usr/local/pgsql/data
```

هناك قيدان يمكن أن يجعلا هذه الطريقة غير عملية:

- يجب إيقاف الخادم قبل حفظ واستعادة البيانات.

- نحتاج لأن نعلم جيداً أننا بهذه الطريقة سنقع في ورطة تذكر أماكن كل الجداول والتحويلات، إلخ. وإلا فيمكن أن ينتهي الأمر بإتلاف قاعدة البيانات. وأيضاً (بشكل عام)، حجم البيانات التي سيتم حفظها بهذه الطريقة سيكون أكبر من تلك التي سنحفظها بالطريقة السابقة، لأنه -على سبيل المثال - بالأمر pg_dump لا يتم حفظ الفهارس، ولكن يتم حفظ أمر إعادة إنشائها.

ملخص تثبيت PostgreSQL:

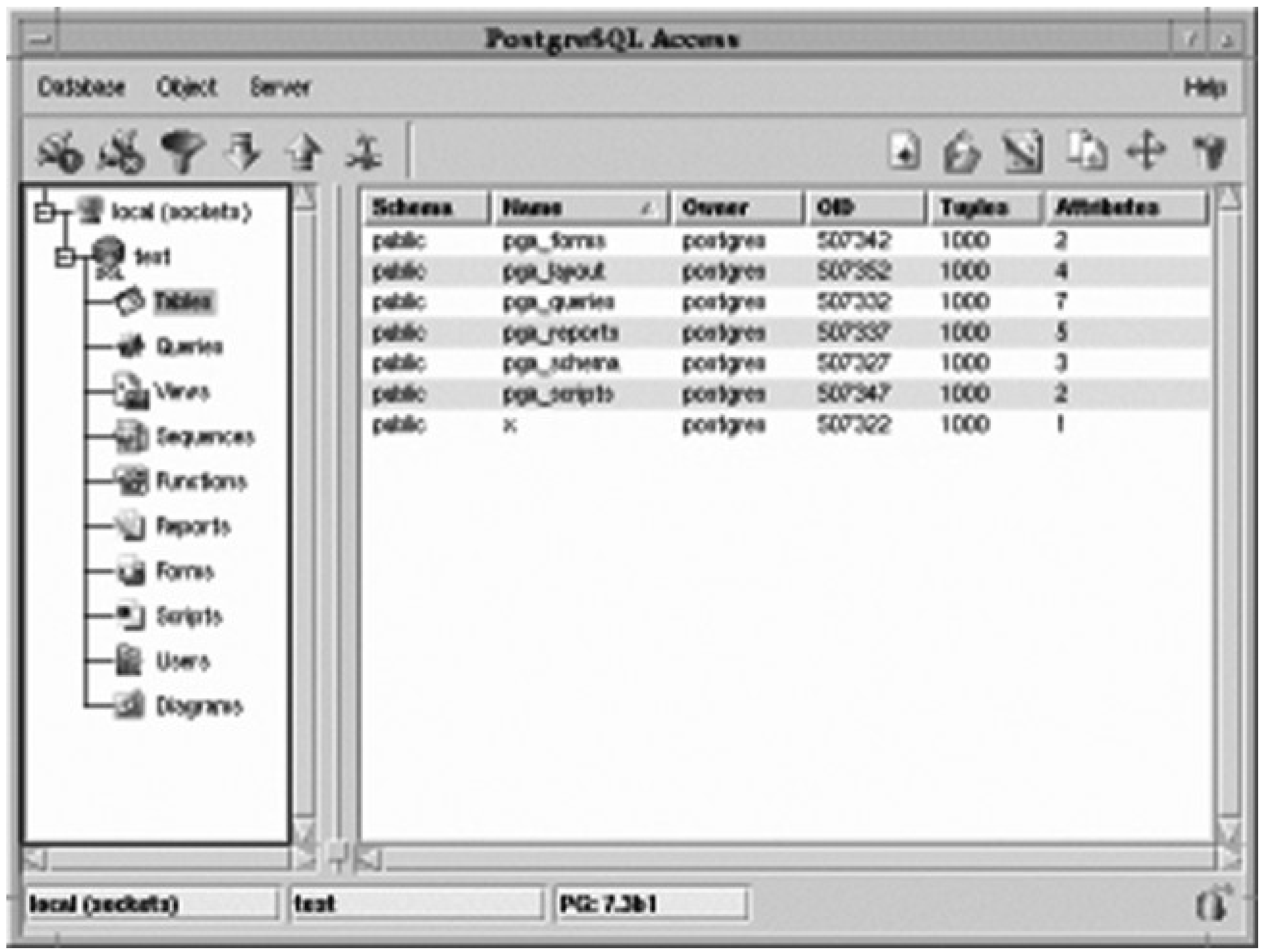
```
./configure
gmake
su
gmake install
adduser postgres
mkdir /usr/local/pgsql/data
chown postgres /usr/local/pgsql/data
su – postgres
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
/usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data > logfile 2>&1 &
/usr/local/pgsql/bin/createdb test
/usr/local/pgsql/bin/psql test
```

1.6 pgaccess

يسمح لنا التطبيق pgaccess (من سطر الأوامر متبوعاً باسم قاعدة البيانات) (http://www.pgaccess.org) بالوصول وإدارة قاعدة البيانات بواجهة رسومية. الطريقة الأسهل للوصول تتم عبر الطرفية، سيكون على مدير قاعدة البيانات أن يقوم بتنفيذ xhost + (إذا لم يكن مستخدم postgresql) مما سيسمح لتطبيقات أخرى أن تتصل بما يعرض لدى المستخدم الحالي.

```
su postgres
pgaccess [DBName] &
```

سيفتح دائماً آخر قاعدة بيانات إذا كان ذلك مضبوطاً في "التفضيلات". يعرض الشكل 1 واجهة pgaccess.



شكل 1: PgAccess

في تثبيت اعتيادي، يمكن للمدير/المستخدم أولاً أن يفتح قاعدة بيانات، ويحدد – كما في هذا المثال – المنفذ = Port 5432، وقاعدة البيانات DataBase = nteum (المعاملات الأخرى ليست ضرورية إذا كانت قاعدة البيانات محلية) ومن ثم الضغط على زر "فتح". كما في هذه اللحظة، سيكون بإمكان المستخدم أن يعمل في مساحة ثنائية الأبعاد وأن يختار ما يريد أن يفعله في المحور الصادي (الجداول، الاستعلامات، العروض، إلخ)، وبإبراز ذاك لعنصر، واختيار واحد من ذلك النوع في النافذة، باستخدام المحور السيني أعلاه لعمل واحد جديد New/Add، أو فتح Open واحد موجود، أو تصميم Design واحد. فإذا اخترنا مثلاً من المحور الصادي Users ومن السيني New، فسيسأل التطبيق عن الاسم، وكلمة المرور (والتحقق)، و timeout، والخصائص (على سبيل المثال، إنشاء قاعدة بيانات، أو إنشاء مستخدمين آخرين). في DataBase يمكننا أيضاً أن نختار التفضيلات، كتغيير نوع الخط – على سبيل المثال – واختيار إمكانية رؤية جداول النظام.

سيتم تسجيل الإعداد الشخصي للمستخدمين في pgaccessrc./~. تساعد الواجهة على عمل/تسهيل كم كبير من عمل المستخدم/المدير، ويُنصح به للمستخدمين حديثي العهد مع PostgreSQL، حيث لن يضطروا لمعرفة سياق سطر الأوامر كما في

psql (سيطلب البرنامج نفسه كل خيارات الأمر عبر نوافذ متعددة).

والأسهل عمل ذلك عبر وحدة webmin ذات العلاقة (سيكون علينا تثبيت الحزم webmin-core والوحدات الضرورية، على سبيل المثال، في هذه الحالة webmin-postgresql)، لكنها غير مضمنة مبدئيًّا في العديد من التوزيعات (لمزيد من المعلومات زُر http://www.webmin.com). أثناء التثبيت، سيُعطي webmin تنبيهاً من أن المستخدم الرئيسي الذي سيُستخدَم هو الجذر، وأنه سيستخدم نفس كلمة مرور المستخدم الجذر لنظام التشغيل. يمكننا عمل ذلك عبر المتصفح على العنوان https://localhost:10000، الذي سيطلب قبول (أو رفض) استخدام شهادة SSL لاتصال SSL، ومن ثم سيظهر كل الخدمات التي يمكن إدارتها، ومن بينها PostgreSQL Data Base Server.

MySQL 2

إن MySQL وفقاً لما يقوله مطوروها هي أكثر قواعد البيانات SQL المفتوحة (وبعبارة أخرى، هي برمجية حرة ومفتوحة المصدر) شعبية، وتملكها حالياً شركة أوراكل. MySQL نظام إدارة قواعد بيانات DBMS. يمكن لـ DBMS أن يضيف ويعالج بيانات مخزنة في قاعدة البيانات. ومثل PostgreSQL، فإن MySQL قاعدة بيانات علاقية، مما يعني أنها تخزن البيانات في جداول بدلاً من حفظها في مكان واحد، مما يوفر سرعة ومرونة أكبر. وبما أنه برنامج حرّ، فيمكن لأيّ كان أن يحصل على المصدر البرمجي وأن يدرسه ويعدله ليوافق احتياجاته، دون الحاجة لدفع شيء من المال، حيث تستخدم MySQL رخصة GPL. يوفر MySQL على صفحته مجموعة من الإحصائيات والخصائص مقارنة مع قواعد البيانات الأخرى، ليري المستخدمين كم هو سريع وكفء وسهل الاستخدام.

2.1 التثبيت

- الحصول عليها من http://www.mysql.com أو أي من مستودعات البرمجيات. يمكن الحصول على كل من الملفات الثنائية والمصدرية للتصريف والتثبيت.

- في حالة الثنائيات، استخدم النسخة المخصصة لدبيان (أو حسب التوزيعة التي لديك)، واختر الحزم *-mysql (الضرورية منها هي common و server و client). بعد السؤال عن بعض الأمور، سينشئ التثبيت مستخدم mysql و مدخلة في etc/init.d/mysql/ لتشغيل وإيقاف الخادم أثناء الإقلاع. يمكن أيضاً عمل ذلك يدوياً باستخدام:

```
/etc/init.d/mysql start|stop
```

- من أجل الوصول إلى قاعدة البيانات، يمكننا ان نستخدم مراقب mysql من سطر الأوامر. إذا حصلنا على الملفات الثنائية (ليس حزم deb أو rpm، ففي تلك الحالة استخدم apt-get أو rpm)، كحزمة gz من موقع mysql، فسيكون علينا أن ننفذ الأوامر التالية من أجل تثبيت قاعدة البيانات:

```
groupadd mysql
useradd -g mysql mysql
cd /usr/local gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
ln -s full-path-to-mysql-VERSION-OS mysql
cd mysql
scripts/mysql_install_db --user=mysql
chown -R root .
chown -R mysql data
chgrp -R mysql .
bin/mysqld_safe --user=mysql &
```

هذا ينشئ مجلد المستخدم/المجموعة، ويفك ضغط قاعدة البيانات ويثبتها في usr/local/mysql/.

- في حالة الحصول على المصدر البرمجي، فالخطوات مشابهة لما يلي:

```
groupadd mysql
useradd -g mysql mysql
gunzip < mysql-VERSION.tar.gz | tar -xvf -
cd mysql-VERSION
./configure --prefix=/usr/local/mysql
make
make install
cp support-files/my-medium.cnf /etc/my.cnf
cd /usr/local/mysql
bin/mysql_install_db --user=mysql
chown -R root .
chown -R mysql var
chgrp -R mysql .
bin/mysqld_safe --user=mysql &
```

من المهم الانتباه عند الضبط، حيث أن prefix = /usr/local/mysql/ و المجلد الذي ستثبت فيه قاعدة البيانات، ويمكن تغييره لإنشاء قاعدة البيانات في أي مكان نريد.

2.2 التحقق وما بعد التثبيت

ما إن يتم التثبيت (سواء عبر الملفات الثنائية أو عبر المصدر البرمجي)، فسيكون علينا التحقق مما إذا كان الخادم يعمل جيداً. في دبيان، يمكن عمل هذا مباشرة:

يشغل الخادم	/etc/init.d/mysql/start
ينشئ معلومات الإصدار	mysqladmin version
يظهر قيم المتغيرات	mysqladmin variables
يطفئ الخادم	mysqladmin -u root shutdown
سيعرض قواعد البيانات المحددة مسبقاً	mysqlshow
سيعرض جداول قاعدة البيانات mysql	mysqlshow sql

إذا كان مثبتاً من المصدر البرمجي، فسيكون علينا قبل القيام بهذه الاختبارات أن ننفذ الأوامر التالية من أجل إنشاء قاعدة بيانات (من مجلد التوزيعة):

```
./scripts/mysql_install_db
cd InstallationDirectoryMysql
./bin/mysqld_safe --user = mysql &
```

إذا كنا نثبت من الحزم الثنائية (RPM, Pkg, ...) فيجب علينا القيام بما يلي:

```
cd InstallationDirectoryMysql
./scripts/mysql_install_db
./bin/mysqld_safe user = mysql &
```

ينشئ النص البرمجي mysql_install_db قاعدة البيانات mysql، ويشغّل mysqld_safe خادم mysqld. ومن ثم، يمكننا فحص كل الأوامر المذكورة أعلاه لدبيان، باستثناء الأمر الأول الذي يشغّل الخادم. إضافة إلى ذلك، إذا كان قد تم تثبيت الاختبارات، فيمكن تشغيلها باستخدام cd sql-bench ثم run-all-tests. ستظهر النتائج في المجلد sql-bench/results للمقارنة مع قواعد البيانات المختلفة.

2.3 برنامج مراقبة MySQL (العميل)

يمكن استخدام عميل MySQL لإنشاء واستخدام قواعد البيانات البسيطة، فهو تفاعلي، ويمكنه الاتصال بالخادم، والقيام بعمليات البحث، وإظهار النتائج. ويعمل أيضاً في الوضع الدفعي (كنصّ برمجي)، حيث يتم تمرير الأوامر إليه عبر ملف. لرؤية كل خيارات الأمر، يمكننا تنفيذ mysql –help. سيكون بإمكاننا عمل اتصال (محلي أو بعيد) باستخدام الأمر mysql، على سبيل المثال، للاتصال عبر واجهة الوب لكن من نفس الجهاز ننفذ:

```
mysql -h localhost -u mysql -p DBName
```

إذا لم ندخل المعامل الأخير، فلن يتم اختيار قاعدة بيانات.

بمجرد أن ندخل، سيظهر mysql محثاً (mysql>) وينتظر أن ندخل أمراً (خاصاً به أو أمر SQL)، كأمر help مثلاً. ومن ثم سننفذ مجموعة من الأوامر لاختبار الخادم (تذكر دائماً أن تضع الفاصلة المنقوطة "؛" في نهاية الأمر):

```
mysql> SELECT VERSION(), CURRENT_DATE;
```

يمكننا أن نستخدم الحروف الكبيرة أو الصغيرة.

```
mysql> SELECT SIN(PI()/4), (4+1)*5; Calculator.
mysql> SELECT VERSION(); SELECT NOW();
```

عدة أوامر في نفس السطر.

```
mysql> SELECT
-> USER()
-> ,
-> CURRENT_DATE;
```

أو في سطور متعددة.

```
mysql> SHOW DATABASES;
```

يعرض الأمر السابق قواعد البيانات المتاحة.

```
mysql> USE test;
```

لتغيير قاعدة البيانات.

```
mysql> CREATE DATABASE nteum; USE nteum;
```

هذا الأمر يُنشئ ويختار قاعدة بيانات اسمها nteum.

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
-> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

يُنشئ جدولاً داخل nteum.

```
mysql> SHOW TABLES;
```

يعرض الجداول و

```
mysql> DESCRIBE pet;
```

يعرض تعريف الجدول.

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

يحمل البيانات من pet.txt في pet. يجب أن يحوي pet سجلاً واحداً في كل سطر مفصولاً بإزاحات اعتماداً على تعريف الجدول (التاريخ بهيئة YYYY-MM-DD[6]).

```
mysql> INSERT INTO pet;
-> VALUES ('Marciano','Estela','gato','f','1999-03-30',NULL);
```

يحمل البيانات التي في السطر.

```
mysql> SELECT * FROM pet;Shows table data.
mysql> UPDATE pet SET birth = "1989-08-31" WHERE name = "Browser";
```

يعدل بيانات الجدول

```
mysql> SELECT * FROM pet WHERE name = "Browser";
```

عينة انتقائية

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
```

عينة مرتبة

6 تبدأ من اليسار بأربع خانات لرقم السنة، تليها شرطة، ثم خانتين للشهر، ثم شرطة، ثم خانتين لليوم.

```
mysql> SELECT name, birth FROM pet WHERE MONTH(birth) = 5;
```

عينة انتقائية مع دوال

```
mysql> GRANT ALL PRIVILEGES ON *.* TO
martian@localhost -> IDENTIFIED BY 'passwd'
WITH GRANT OPTION;
```

إنشاء المستخدم marciano في قاعدة البيانات. يجب أن يتم تنفيذ هذا بالمستخدم الجذر لقاعدة البيانات. أو يمكن أن يتم هذا مباشرة باستخدام:

```
mysql> INSERT INTO user (Host,User,Password) ->
VALUES('localhost','marciano','passwd');
```

2.4 الإدارة

لـ MySQL ملف إعداد في /etc/mysql/my.cnf (في دبيان) حيث يمكن تغيير الخيارات المبدئية لقاعدة البيانات، والتي تشمل: منفذ الاتصال، المستخدم، كلمة مرور المستخدمين البعيدين، ملفات التقارير، ملفات البيانات، وفيما إذا كان يقبل اتصالات خارجية، إلخ. فيما يتعلق بالأمن، علينا أخذ بعض الاحتياطات:

١) عدم إعطاء أي أحد (ما عدا المستخدم الجذر لـ MySQL) وصولاً إلى جدول المستخدمين في قاعدة بيانات MySQL، حيث تكمن هناك كلمات مرور المستخدمين، والتي يمكن أن تستخدم لأغراض أخرى.

٢) التحقق من mysql -u root. إذا تمكنا من الوصول، فهذا يعني أنه لا توجد كلمة مرور للمستخدم الجذر. لتغيير هذا، يمكننا أن نقوم بما يلي:

```
mysql -u root mysql
mysql> UPDATE user SET Password = PASSWORD('new_password')
-> WHERE user = 'root';
mysql> FLUSH PRIVILEGES;
```

الآن، للاتصال بالمستخدم الجذر:

```
mysql -u root -p mysql
```

٣) تحقق من التوثيق المتعلق بالظروف الأمنية وبيئة الشبكة لتفادي المشاكل المتعلقة بالهجمات و/أو الاختراقات.

٤) لعمل نسخ من قواعد البيانات، يمكننا أن نستخدم الأوامر التالية:

```
mysqldump --tab = /DestinationDirectory \
--opt DBName
```

أو

```
mysqlhotcopy DBName /DestinationDirectory
```

وكذلك، يمكننا أن ننسخ الملفات frm.* و MYD.* و MYI.* عند توقف الخادم. ولاستعادتها نفّذ:

```
REPAIR TABLE o myisamchk -r
```

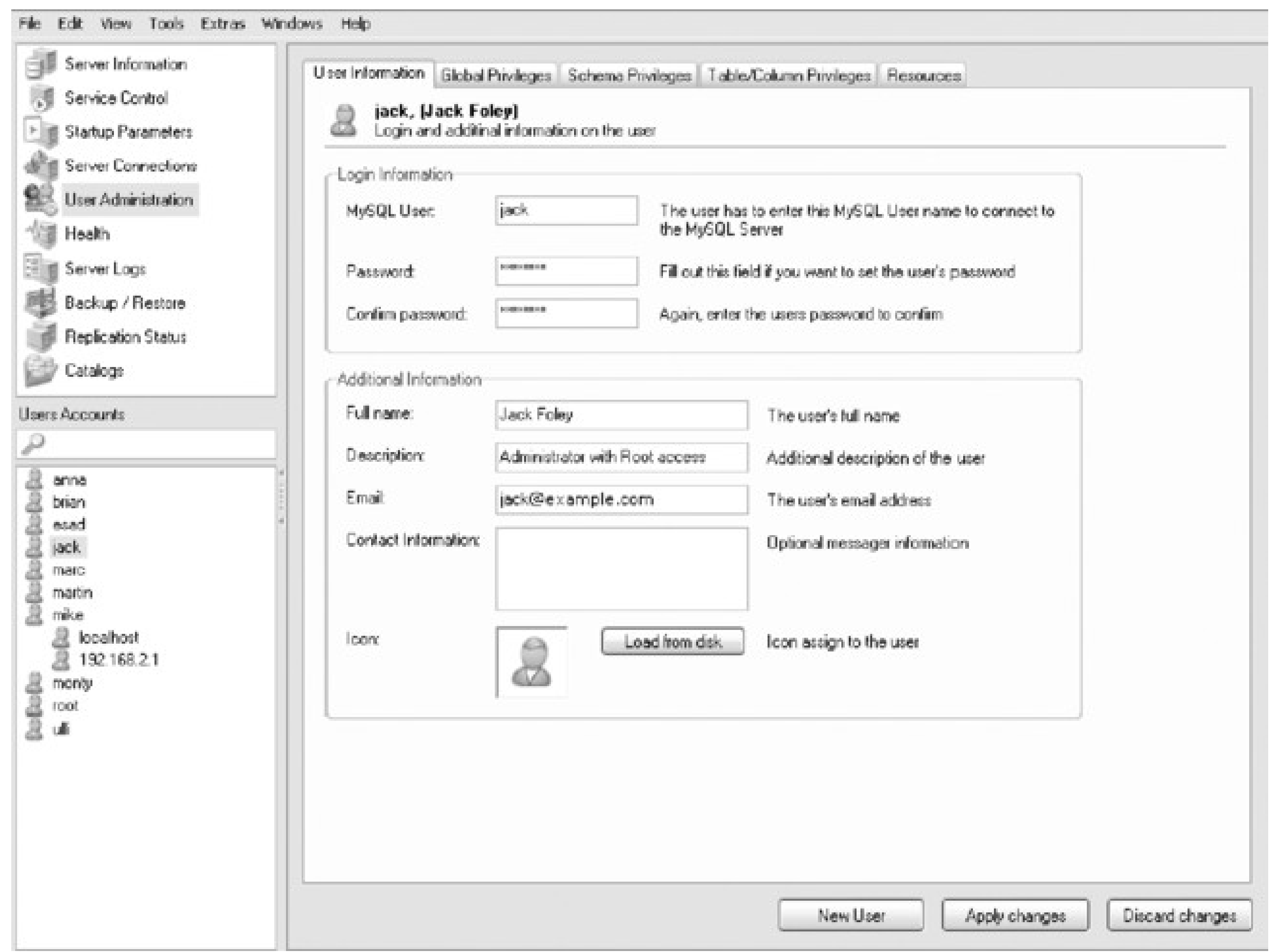
والتي ستعمل في 99% من الحالات. عدا ذلك، يمكننا أن ننسخ الملفات المحفوظة ونشغل الخادم. هناك طرق أخرى بديلة تعتمد على ما نرغب باسترجاعه، كإمكانية حفظ/استرجاع جزء من قاعدة البيانات (راجع النقطة 4,4 من التوثيق).

2.5 الواجهات الرسومية

هناك عدد كبير من الواجهات الرسومية لـ MySQL، ومن ضمنها MySQL Administrator (يمكن الحصول عليها من http://www.mysql.com/products/tools/administrator)، و MySQL-Navigator (من http://sourceforge.net/projects/mysqlnavigator)، و Webmin مع الوحدة التي تعمل مع MySQL (الحزم webmin-core و webmin-mysql)، رغم أن هذه الأخيرة لم تعد مضمنة في بعض التوزيعات. كما في PostgreSQL،

تسمح Webmin بالعمل مع MySQL (سنحتاج لتثبيت الحزمتين المذكورتين). أثناء التثبيت، ستنبه webmin إلى أن المستخدم الرئيسي سيكون الجذر root وأنه سيستخدم نفس كلمة مرور الجذر لنظام التشغيل. للاتصال بـ Webmin علينا أن نطلب – على سبيل المثال – http://localhost:10000 في شريط العنوان للمتصفح الذي سيطلب قبول (أو رفض) استخدام شهادة اتصال SSL، ومن ثم ستظهر كل الخدمات التي يمكن إدارتها، ومن بينها خادم قواعد البيانات MySQL Data Base Server.

إن MySQL Administrator تطبيق قوي لإدارة والتحكم بقواعد البيانات المعتمدة على MySQL. يدمج هذا التطبيق بين إدارة قواعد البيانات، والتحكم بها، وصيانتها، بشكل سهل وفي نفس البيئة. خصائصه الأساسية هي: إدارة متقدمة لقواعد البيانات الضخمة، وأخطاء أقل عبر "الإدارة المرئية"، وإنتاجية أكبر، إضافة إلى كونها بيئة إدارة آمنة. الشكل التالي يعرض مشهداً من MySQL Administrator (يمكننا أن نجد كل التوثيق المتعلق بتثبيته وتشغيله في http://dev.mysql.com/doc/administrator/en/index.html).



شكل 2: MySQL Administrator

3 برامج إدارة المصادر البرمجية[7]

إن برامج إدارة المصادر البرمجية أو برامج إدارة الإصدارات هي تطبيقات تسمح بإدارة وصيانة إصدارات الوثائق، وبرامج الحاسوب والمواقع الكبيرة، وغيرها من الملفات (وغالباً ما تستخدم لإدارة المصادر البرمجية)، محتفظة بسجل عن عمليات التعديل، ومن قام بها، ومتى، ولماذا. ومن الأمثلة عليها: RCS, CVS, SVN, HQ, Git، وسنتعرض في هذا الجزء لشرح svn لما له من انتشار واسع، أما CVS (المبني على RCS) الذي كان شائعاً جدًّا في ما مضى، فقد تراجع استخدامه كثيراً لصالح هذين الإثنين، ولهذا فلن نتطرق إليه في هذا الكتاب. يمكن لمن يريد شرحاً لنظام CVS أن يراجع "تنصيب وإعداد مخدم CVS لإدارة الشفرة المصدرية" على الرابط التالي: http://www.infomag.news.sy/index.php?inc=issues/showarticle&issuenb=28&id=578، أو "CVS and Subversion: Combined Tutorial" لشرح كل من CVS و SVN (بالإنجليزية) على الرابط التالي:

http://www.developingprogrammers.com/index.php/2005/11/24/cvs-and-subversion-combined-tutorial.

كما سبق وذكرنا في المقدمة، فإن Subversion (الموقع http://subversion.tigris.org) هو برمجية نظام تحكم بالإصدارات صمم خصيصاً ليحل محل CVS، ولزيادة إمكانياته. وهو برمجية حرة متاحة تحت رخصة من نوع Apache/BSD، ويعرف أيضاً بالاسم svn، وهو تطبيق سطر الأوامر الخاص به. من مزاياه الهامة أنه على النقيض من CVS، فالملفات ليس لكل منها رقم مراجعة مستقل عن البقية، بل هناك رقم مراجعة واحد للمستودع بأكمله يحدد الحالة المشتركة لكل ملفات المستودع في الوقت الذي أخذت فيه هذا الرقم. ومن المزايا الأساسية يمكننا أن نذكر أيضاً:

- يمكن تتبع تاريخ الملف والمجلد عبر النسخ الاحتياطية وإعادة التسمية.
- تعديلات جزئية وآمنة (بما فيها تعديلات على ملفات عديدة).

7 لقد خضعت ترجمة هذا الجزء لتعديلات كبيرة، كما تم حذف الجزء المتعلق بشرح CVS، وإضافة روابط لشروحات خارجية لإثرائه، إلخ. يمكن للمهتم الرجوع إلى الكتاب الأصل على الموقع http://ftacademy.org/materials/fsm/2#1.

- إنشاء الفروع والتسميات بسهولة وكفاءة.

- يتم إرسال الاختلافات فقط في كلي الاتجاهين (في CVS يتم دائماً إرسال الملفات الكاملة إلى الخادم).

- يمكن أن يُخدَّم في أباتشي، عبر WebDAV/DeltaV.

- يتعامل مع الملفات الثنائية بكفاءة (عكس CVS الذي يتعامل معها داخلياً كما لو كانت نصوص).

هناك كتاب مجاني مثير للاهتمام يشرح كل ما يتعلق بـ Subversion (على العنوان http://svnbook.red-bean.com/index.html)، وهو متوفر بعدة لغات، لكن العربية ليست منها.

وأما بخصوص برمجية Git اللامركزية لإدارة المصادر البرمجية، والتي أنشأها لينُس تورفَلدز لكي يستخدمها لإدارة المصادر البرمجية للنواة، والتي نشرها تحت رخصة جنو العامة GNU GPL v2، فقد ازدادت شعبيتها كثيراً خلال الأعوام الماضية، وتبنتها مشاريع عديدة. إذا كنت ترغب بالاطلاع أكثر على برمجية Git، فألقِ نظرة على الشرح التالي من مشروع أعجوبة: http://www.ojuba.org/wiki/docs/git_tutorial.

4 Subversion

كفكرة ابتدائية، يساعد Subversion على إدارة مجموعة من الملفات (مستودع) وإصداراتها المختلفة. ونذكر أيضاً أنه لا يهمنا كيفية حفظ الملفات، ولكن كيف يمكننا الوصول إليها. فكرة المستودع تشبه مجلداً يمكننا أن نستعيد منه ملفاً سواء كان عمره أسبوعاً واحداً أو 10 أشهر اعتماداً على حالة قاعدة البيانات، للحصول على الإصدارات الأخيرة، وإضافة إصدارات جديدة. على النقيض من CVS، يعمل Subversion مراجعات عامة للمستودع، مما يعني أن تعديلاً في ملف ما لن يغير إصدار ذلك الملف وحده، بل بزيادة رقم مراجعة المستودع بأكمله بقيمة واحد. إضافة إلى الكتاب الذي ذكرناه (http://svnbook.red-beans.com/nightly/en/index.html)، راجع التوثيق في: http://subversion.tigris.org/servlets/ProjectDocumentList.

في دبيان، علينا استخدام apt-get install subversion، وإذا كنا نرغب بنشر المستودعات في أباتشي 2، فسنحتاج أيضاً apt-get install apache2-common والوحدة الخاصة بذلك apt-get install libapache2-subversion.

- الخطوة الأولى: إنشاء مستودعنا، والمستخدم (لنفترض أن المستخدم هو svuser)، والمجموعة (svgroup)، ونقوم ذلك بصلاحيات الجذر:

```
mkdir -p /usr/local/svn
addgroup svgroup
chown -R root.svgroup /usr/local/svn
chmod 2775 /usr/local/svn
```

- إضافة المستخدم svuser إلى المجموعة svgroup بالأمر addgroup svuser svgroup.
- نتصل بالمستخدم svuser ونتأكد أننا في المجموعة svgroup بالأمر groups.
- Svnadmin create /usr/local/svn/tests

سيُنشئ هذا الأمر مجموعة من الملفات والمجلدات لإدارة الإصدارات والتحكم بها. إذا لم يكن مصرّحاً لنا في /usr/local/svn، فيمكننا القيام بذلك في المجلد المحلي: mkdir -p $HOME/svndir ومن ثم svnadmin create

.$HOME/svndir/tests

- ثم ننشئ مجلداً مؤقتاً mkdir -p $HOME/svntmp/tests، ثم ننتقل إلى المجلد cd $HOME/svntmp/tests وأنشئ ملفاً مثل: echo First File Svn 'date' > file1.txt.

- ننقله إلى المستودع: بداخل المجلد نكتب "svn import file:///home/svnuser/svndir/tests -m "view initial. إذا كنا أنشأناه في usr/local/svn/tests/، فعلينا كتابة المسار الكامل بعد //:file. ينسخ الأمر import شجرة المجلدات، ويسمح الخيار m- بعرض رسالة الإصدار. إذا لم نضف الخيار m، فسيفتح محرر لعمل ذلك (سنحتاج لإضافة رسالة لتفادي المشاكل). المجلد الفرعي $HOME/svntmp/tests نسخة من العمل في المستودع، ويُنصح بحذفه حتى لا تقع في الخطأ بالتعديل عليه بدلاً من المستودع[8] (rm -rf $HOME/svntmp/tests)[9].

- بمجرد الدخول إلى المستودع، يمكننا الحصول على النسخة المحلية حيث يمكننا العمل ومن ثم رفع النسخ إلى المستودعات، وذلك كالآتي:

```
mkdir $HOME/svm-work
cd $HOME/svn-work
svn checkout file:///home/svuser/svndir/tests
```

حيث سنجد أنه صار لدينا المجلد tests. يمكننا أن ننسخ باسم آخر بإضافة الاسم الذي نريده في النهاية. لإضافة ملف جديد إليه:

```
cd /home/kikov/svn-work/tests
echo Second File Svn 'date' > file2.txt
svn add file2.txt
svn commit -m"New file"
```

من المهم أن نلاحظ أنه بمجرد وجودنا في النسخة المحلية (svn-work) فعلينا أن لا نحدد المسار. يضيف svn علامات لإضافة الملف إلى المستودع وأنه في الحقيقة يتم إضافته عندما ننفذ svn commit. سيعطينا بعض الرسائل

---

8 وجود أكثر من ملف بنفس الاسم والصفات قد يصعب عليك التمييز بينها، فتظن أحدها الآخر.

9 كن حذراً عند استخدام الخيارين r و f مع الامر rm، فخطأ بسيط قد يسبب فقداناً لبيانات هامة!

التي تشير إلى أنه في إصداره الثاني.

إذا أضفنا سطراً جديداً إلى الملف file1.txt بالأمر echo 'date' >> file1.txt، فسيكون بإمكاننا رفع التعديلات بالأمر: "svn commit -m "New Line.

يمكن مقارنة الملف المحلي بملف المستودع، فعلى سبيل المثال، نضيف سطراً ثالثاً إلى file1.txt بالأمر 'echo 'date file1.txt، ولكن لا نرفعه، وإذا كنا نرغب برؤية الاختلافات ننفّذ: svn diff.

سيبرز هذا الأمر الاختلافات بين الملف المحلي وملف المستودع. إذا رفعناه بالأمر "svn commit -m "new line2 (الذي سيُنشئ إصداراً آخر) فلن يعطينا svn diff أيّة اختلافات.

يمكننا أيضاً استخدام الأمر svn update داخل المجلد لتحديث النسخة المحلية. إذا كان هناك مستخدمان أو أكثر يعملان في نفس الوقت، وعمل كل منهما نسخة محلية عن المستودع وعدّله (بتنفيذ commit)، فعندما يقوم المستخدم الآخر بإيداع نسخة بتعديلاته، فسيتلقى رسالة خطأ تعارض، حيث أن النسخة التي في المستودع لها تاريخ تعديل أحدث من النسخة الأصل التي عدل عليها هذا المستخدم (وبعبارة خرى، حدثت تعديلات فيما بين النسختين)، مما يعني أنه إذا قام المستخدم الثاني بالإيداع (commit)، فسنخسر تعديلات الأول. لعمل ذلك، علينا أن ننفذ svn update الذي يخبرنا بالملف الذي يتسبب بوجود تعارض (conflict) بالإشارة إليه بحرف C، وسيرينا الملفات التي تكمن فيها الأجزاء المتعارضة. يجب على المستخدم أن يقرر أي الإصدارات يبقي، وما إذا كان بإمكانه أن يودِع.

ومن الأوامر المفيدة svn log file1.txt الذي سيظهر كل التغييرات التي طرأت على الملف والإصدارات المتعلقة به.

وهناك ميزة هامة في subversion وهي أنه بإمكانه العمل بالتكامل مع apache2 (وأيضاً على SSL) ليتم الوصول إليه من جهاز آخر (راجع العملاء في http://svnbook.red-bean.com) أو ببساطة انظر إلى المستودع. في Debian Administration هناك شرح لكيفية ضبط Apache2 و SSL، أو كما أشرنا سابقاً في الجزء المتعلق بالخوادم. من أجل هذا علينا تفعيل الوحدة WebDAV (انظر إلى http://www.debian-administration.org/articles/285 أو في غيابها http://www.debian-administration.org/articles/208.

كمستخدم جذر ننفذ:

```
mkdir /subversion chmod www-data:www-data
```

حتى يتمكن أباتشي من الوصول إلى المجلد

```
svnadmin create /subversion
```

ننشئ المستودع

```
ls -s /subversion

-rw-r--r--    1    www-data    www-data    376     May 11 20:27 README.txt
drwxr-xr-x    2    www-data    www-data    4096    May 11 20:27 conf
drwxr-xr-x    2    www-data    www-data    4096    May 11 20:27 dav
drwxr-xr-x    2    www-data    www-data    4096    May 11 20:28 db
-rw-r--r--    1    www-data    www-data    2       May 11 20:27 format
drwxr-xr-x    2    www-data    www-data    4096    May 11 20:27 hooks
drwxr-xr-x    2    www-data    www-data    4096    May 11 20:27 locks
```

للاستيثاق نستخدم htpasswd (على سبيل المثال بالأمر:

```
htpasswd2 -c -m /subversion/.dav_svn.passwd user
```

المُنشأ بالمستخدم www-data. سيكون علينا أن نكتب الـ c- في المرة الأولى التي ننفذ فيها الأمر لإنشاء الملف. يخبرنا هذا أنه من أجل الوصول إلى هذا المجلد، نحتاج لكلمة مرور (وهي كلمة المرور التي أدخلناها للمستخدم).

ثم سنحتاج لتغيير الملف httpd.conf بحيث يصير شبيهاً بما يلي:

```
<location /svn>
        DAV svn
        SVNPath /subversion
        AuthType Basic
        AuthName "Subversion Repository"
        AuthUserFile /subversion/.dav_svn.passwd
        Require valid-user
</location>
```

نعيد تشغيل أباتشي، ونحن الآن مستعدون لاستيراد بعض الملفات، مثل:

```
svn import file1.txt http://url-server.org/svn \ -m "Import Initial"
```

سيُطلب منا الاستيثاق (بالمستخدم وكلمة المرور) وسيتم إخبارنا بأن الملف file1.txt قد تمت إضافته إلى المستودع.

الأنشطة

١) عرف في PostgreSQL قاعدة بيانات فيها على الأقل 3 جداول كل منها ذو 5 أعمدة (بحيث يكون 3 من هذه الأعمدة الخمسة رقمياً).

أنشئ قائمة مرتبة لكل جدول/عمود. أنشئ قائمة مرتبة حسب أكبر قيمة في عمود ما (العمود س) من كل الجداول. غير القيمة العددية في عمود آخر (العمود ص) إلى القيمة العددية الناتجة من جمع قيمة عمود آخر (العمود ع) وقيمة ((العمود ل)/2).

٢) نفس التمرين السابق، لكن مع MySQL.

٣) اضبط Subversion لعمل 3 مراجعات لمجلد حيث هناك 4 ملفات بلغة سي و makefile. اعمل فرعاً للملف ثم ادمجه في الفرع الرئيسيّ.

٤) حاكِ الاستخدام المتزامن لملف في طرفيتي لينكس واذكر تسلسل الخطوات التي يجب عملها بحيث ينعكس التغييران المختلفان للمستخدمَين في مستودع svn.

٥) نفس التمرين السابق، لكن يجب أن يتصل أحد المستخدمين من جهاز آخر.

المراجع

مصادر أخرى للمراجع والمعلومات:

[Debc, Ibi, Mou01]

PgAccess: http://www.pgaccess.org/

WebMin: http://www.webmin.com/

Mysql Administrator: http://www.mysql.com/products/tools/administrator/

Apache2 and SSL: http://www.debian-administration.org/articles/349

Apache2 and WebDav: http://www.debian-administration.org/articles/285

Subversion: http://subversion.tigris.org

كتاب مجاني عن Subversion هنا: http://svnbook.red-bean.com/index.es.html

هناك كم كبير من التوثيق عن Apache & SSL + Subversion في دبيان، إضافة إلى http://www.debian-administration.org. اكتب في جوجل "Apache2 SSL and Subversion" للحصول على بعض التوثيق المثير للاهتمام.

مراجع عربية:

مقالة الأستاذ مؤيد السعدي "أساسيات نظام إدارة إصدارات المصدر git" على موقع أعجوبة:

http://www.ojuba.org/wiki/docs/git_tutorial

إدارة الأمن

د. جُسِـــپ جُبرا إستيڤَ

مقدمة

إن القفزة التقنية من أنظمة الحواسيب المكتبية المعزولة إلى الأنظمة الحالية الموجهة نحو الشبكات المحلية والإنترنت قد أضافت عقبة جديدة أمام المهام الاعتيادية للمدير: التحكم بأمن النظام. الأمن مجال معقد يتضمن تقنيات تحليلية مع تقنيات لاكتشاف ومنع الهجمات المحتملة، كتحليل العوامل "النفسية"، وما يتعلق بسلوك مستخدمي النظام والنوايا المحتملة للمهاجمين.

يمكن أن تأتي الهجمات من مصادر عديدة، ويمكن أن يتراوح تأثيرها من تطبيق أو خدمة أو مستخدم بعينه إلى كل هذه معاً، بل وحتى النظام بأكمله.

يمكن أن تغير الهجمات المحتملة سلوك الأنظمة أو حتى أن تجعلها تنهار (تعطيلها)، أو إعطاء شعور خاطئ بالأمن والذي يمكن أن يكون اكتشافه صعباً. يمكننا المرور على هجمات الاستيثاق (الحصول على وصول عبر البرامج أو المستخدمين المعطلين سابقاً)، أو الاعتراض (إعادة توجيه أو اعتراض قنوات الاتصال وحركة البيانات فيها)، أو الإبدال (إحلال تطبيقات أو مستخدمين أو أجهزة محل أخرى دون أن يتم ملاحظة التغييرات).

علينا أن نبقي في بالنا أنه يستحيل الحصول على أمن كامل 100%.

تقنيات الأمن سلاح ذو حدين يمكنه ببساطة أن يعطينا شعوراً خاطئاً بالسيطرة على المشكلة. الأمن هذه الأيام قضية شائكة وشاملة، والأهم من ذلك أنها متغيرة أيضاً. لا يمكننا أبداً أن نتوقع أو نقول بأن الأمن مضمون، بل سيكون على الأغلب إحدى النواحي التي يجب على المدراء أن يقضوا معظم الوقت معها، وأن تبقى المعلومات المتعلقة بها محدّثة.

سنبحث في هذه الوحدة في بعض أنواع الهجمات التي يمكن أن نواجهها، وكيف يمكننا التحقق من الأمر، وكيف نمنع أجزاء من الأمن المحلي وبيئة الشبكة من أن يتم مهاجمتها. وإضافة إلى ذلك، سنختبر تقنيات لاكتشاف الاختراقات وبعض الأدوات الأساسية التي يمكن أن تساعدنا في التحكم بالأمن.

علينا أن نذكر أيضاً أنه يمكننا في هذه الوحدة أن نعرض فقط بعض النواحي المتعلقة بالأمن هذه الأيام. لأي نوع من التعلم المتخصص، ننصح بتفقّد المراجع المتاحة، إضافة إلى أدلة استخدام المنتجات والأدوات التي تم تغطيتها ههنا.

1 أنواع وأساليب الهجوم

يمكن فهم أمن الحاسوب من وجهة نظر مدير النظام على أنها العملية التي تسمح لمدير النظام أن يمنع ويكتشف الاستخدام غير المخوّل للنظام. تساعد إجراءات الحماية على منع محاولات المستخدمين غير المخولين (المعروفين بالدُخلاء) من الوصول إلى أي جزء من النظام. يساعد الاكتشاف على كشف وقت هذه المحاولات، أو – إذا كانت فعّالة – إنشاء حواجز بحيث لا يتم تكرار هذه الاختراقات وبحيث يتم استعادة النظام إذا تم اختراقه.

يطمح الدُخلاء (والذين يُعرفون أيضاً بأسماء أخرى كالمخترقين، وكاسري الحماية، والقراصنة، والهاكرز[1]، والكراكرز، وغيرها) عادة للحصول على تحكم بالنظام، إما للتسبب في خلل فيه، أو لإتلاف النظام أو بياناته، أو لاستخدام موارد الجهاز، أو ببساطة لاستخدامه لشنّ هجمات ضد أنظمة أخرى، مما يساعدهم على حماية هويتهم وإخفاء المصدر الأصلي للهجوم. ويمكن أيضاً أن يكون هدفهم معرفة (أو سرقة) معلومات النظام، أو التجسس المباشر على مهام النظام، أو التسبب بتلف ماديّ للجهاز، بتهيئة القرص، أو تغيير البيانات، أو حذف أو تعديل برمجيات حيوية، إلخ.

فيما يتعلق بالمخترقين، علينا أن نعرف بعض الاختلافات غير الواضحة للعموم. بالعادة، يشار إلى الهاكر Hacker إلى الشخص الخبير ذي المعلومات المفصلة بما يتعلق بالحاسوب، والذي لديه شغف بالبرمجة وقضايا الأمن، والذي يستخدم معلوماته – دون أهداف خبيثة – لحماية نفسه أو غيره، وبالدخول إلى الشبكات لاكتشاف المشاكل الأمنية، وفي بعض الحالات لاختبار قدراتهم.

---

1 إن كلمة هاكر hacker (وتجمع هاكرز، وهي دارجة بالعربية: هكر، وهكرز) لها دلالة أخرى، وتعني من لديهم خبرة كبيرة في مجال الحاسوب برمجة وعتاداً وما إلى ذلك، وكثير منهم مهتمون بمجال الأمن، وغالباً ما يقوم هؤلاء باكتشاف الثغرات وإصلاحها، ويطلق عليهم البعض بالعربية لفظ "الخارقين"، كبديل عن "المخترقين"، أما "المخترقون" فيطلق عليهم هؤلاء كلمة كراكرز crackers، لكن المعنى بدأ يأخذ منحنىً آخر مع الوقت، فصار يدل على من لديهم خبرة في مجال أمن المعلومات، ومن يقومون بالاختراق، وتم تقسيمهم إلى ثلاثة أقسام: ذوي القبعات البيضاء – الذين يكتشفون الثغرات ويصلحونها –، وذوي القبعات السوداء – ذوي النوايا الخبيثة –، وذوي القبعات الرمادية – وهم من يدمجون بين الإثنين –. وهناك نقطة اخرى، وهي الـ hacktivism، أي الاختراق دفاعاً عن قضايا معينة سياسية كانت أو بيئية أو إنسانية ... إلخ. تفقد المراجع للمزيد حول الموضوع. وهناك أيضاً مصطلح الإرهاب التقني cyber-terrorism ... إلخ.

كمثال على ذلك مجتمع جنو/لينكس، والذي يدين له الهاكرز الذين فيه بالكثير، حيث يجب أن يتم فهم كلمة "هاكر" نكبير في نواحٍ معينة (وليس كمتسلل أو مخترق للأمن).

وفي نفس الوقت، لدينا كاسروا الحماية (أو المخترقون) crackers. وتستخدم هذه الكلمة بشكل سلبي للإشارة إلى من يستخدمون معرفتهم من أجل تخريب (أو إتلاف) أنظمة، سواء كان هذا من أجل الشهرة، أو لأسباب مالية، وسواء كانوا ينوون التسبب بتلف أو فقط إزعاج، لأسباب تتعلق بالتجسس التقني، أو عمليات الإرهاب التقني، إلخ. وكذلك نستخدم كلمتي hacking و cracking عندما نشير إلى تقنيات دراسة أو اكتشاف أو حماية الأمن، أو – على العكس – التقنيات المصممة للتسبب بأضرار باختراق أمن أنظمة.

لسوء الحظ، الحصول على وصول إلى النظام (سواء كان غير محمي أو آمن جزئيًّا) أسهل بكثير مما قد يبدو. يكتشف الدُخلاء باستمرار ثغرات vulnerabilities جديدة (تعرف أحياناً بالثقوب holes أو الاستغلالات exploits)، والتي تسمح لهم بدخول طبقات مختلفة من البرمجيات. إن التعقيد المتزايد والذي لا ينتهي للبرمجيات (والعتاد) يجعل اختبار أمن أنظمة الحاسوب بطريقة معقولة أصعب وأصعب. إن الاستخدام الشائع لجنو/لينكس في الشبكات – سواء الإنترنت أو الشبكات الخاصة المعتمدة على تقنيات TCP/IP، كالإنترانت – تجعلنا كضحايا نكشف أنظمتنا للهجمات.

إن أول ما علينا فعله هو كسر خرافة الأمن المطلق للحاسوب: إنها ببساطة غير موجودة. إن ما يمكننا تحقيقه هو مستوى معين من الأمن يجعلنا نشعر بالأمان في ظل معطيات معينة. ولكن كذلك، هذا مجرد فهم للأمن، وكأيّ فهم يمكن أن يكون خاطئاً، بحيث يمكن أن نصير منتبهين في اللحظة لأخيرة والتي يصير فيها نظامنا متأثراً بالفعل. إن الاستنتاج المنطقي هو أن أمن الحاسوب يتطلب جهداً هامًّا فيما يتعلق بالاستقرار والواقعية والتعلم بشكل يوميّ.

علينا أن نكون قادرين على عمل سياسات أمنية لأنظمتنا تسمح لنا بمنع الهجمات المحتملة، والتعرف والردّ عليها. وأن نكون واعين لأن الشعور بالأمن الذي قد يكون لدينا لا يتخطى كونه "شعوراً". ولهذا، علينا أن لا نتجاهل أيّة سياسات متبعة، ونحتاج لأن نبقيها محدّثة، مثلها في ذلك مثل معلوماتنا.

إن الهجمات المحتملة خطر محدِق بأنظمتنا، ويمكن أن يؤثر على أدائها، إضافة إلى البيانات التي نتعامل معها؛ سيكون

علينا دائماً أن نحدد سياسة معينة لمتطلبات الأمن لأنظمتنا وبياناتنا. التهديدات التي يمكن أن نعاني منها يمكن أن تؤثر على النواحي التالي:

- **السرّيّة Confidentiality:** يجب أن تكون البيانات متاحة فقط للأشخاص المخوّلين، نحن نجيب عن السؤال: من سيكون بإمكانه الوصول إليها؟
- **الصحّة Integrity:** يجب أن يعدل البيانات فقط الأشخاص المخول لهم بذلك: ما الذي يمكن عمله بهذا الخصوص؟
- **الإتاحة Availability:** يجب أن تكون المعلومات متاحة لمن يحتاجونها وقتما يحتاجونها في حال كانوا مخولين بذلك: كيف ومتى يمكن الوصول إليها؟

فلننتقل إلى تصنيف معين (غير مفصّل) للأنواع الاعتيادية من الهجمات التي يمكن أن نعاني منها:

- **الاستيثاق:** الهجمات التي تزوّر هوية عضو بحيث يتم الحصول على وصول إلى برامج أو خدمات لم تكن متاحة قبل ذلك.
- **الاعتراض (أو التنصت):** آليات يتم بها اعتراض البيانات على يد آخرين لم تكن موجهة إليهم.
- **التغيير Falsification** (أو الإبدال): إبدال بعض العناصر - سواء كانت أجهزة أو برمجيات أو بيانات - إلى أخرى خاطئة.
- **سرقة الموارد:** استخدام غير مخول للموارد.
- **أو ببساطة التخريب:** وعلى كل حال، وجود آليات تسمح بالتدخل في العمل الصحيح للنظام أو الخدمات للتسبب بإزعاج في جزء ما، أو إيقاف تشغيل أو إلغاء لموارد أمرٌ شائع نوعاً ما.

الطريقة والتقنيات المحدّدة المستخدمة يمكن أن تتنوع بشكل كبير (إضافة إلى ذلك، تظهر ابتكارات في هذا المجال يومياً)، مما يجبرنا - كمدراء - أن نبقى على تواصل دائم مع المجال الأمني لمعرفة ما يمكن أن نواجه يوميًّا. وفيما يتعلق بكل نوع

من أنواع الهجوم هذه، يمكن في العادة استخدام طريقة واحدة أو أكثر، والتي يمكن بدورها أن تُحدِث نوعاً أو اكثر من الهجوم.

وفيما يتعلق بالمكان الذي يمكن حدوث الهجوم فيه، نحتاج لأن نوضح ما يمكن أن يتم عمله وما يمكن أن تكون أهداف هذه الأساليب[2]:

- العتاد: من هذه الناحية، الخطر يتعلق مباشرة بالإتاحة، ماذا يمكن لشخص لديه وصول إلى العتاد أن يفعل؟ في هذه الحالة، سنحتاج في الوضع الطبيعي إلى إجراءات "فيزيائيّة"، كالتحكم بالوصول إلى الأماكن التي تقبع فيها هذه الأجهزة من أجل منع مشكلات سرقة أو تلف الأجهزة والذي يسبب إزالة خدمتها. يمكن أن تتأثر أيضاً السرية والصحة إذا كان الوصول إلفيزيائيّ إلى الأجهزة يسمح لبعض أجهزتها - كالأقراص الصلبة - بأن تُستخدَم، أو إذا كانت تسمح بالإقلاع بالأجهزة أو الوصول إلى حسابات مستخدمين يمكن أن تكون مفتوحة.

- البرمجيات: إذا تم التأثير على الوصول أثناء الهجوم، فيمكن أن تكون البرامج محذوفة أو معطلة مما يمنع الوصول إليها. في حالة السرية، يمكن أن تتسبب بنسخ غير مصرح به للبرمجيات. في حالة الصحّة، فإن العمل الأساسي للبرنامج يمكن أن يتم تغييره، بحيث تفشل في ظروف معينة، أو بحيث تقوم بمهام تفيد المهاجم، أو يمكن ببساطة أن تؤثر على صحة بيانات البرنامج: بجعلها عامة، أو تغييرها، أو ببساطة سرقتها.

- البيانات: سواء كانت ذات هيكلية معينة، كما في خدمات قواعد البيانات أو برامج إدارة الإصدارات، أو ملفات بسيطة. الهجمات التي تهدد الوصول يمكن أن تدمر أو تمحق هذه البيانات، وبالتالي منع الوصول إليها. أما في حالة السرية، فيمكن أن نسمح بالقراءة غير المخولة، ويمكن أن تتأثر الصحة عندما يتم عمل تعديلات أو إنشاء بيانات جديدة.

- قناة الوصول (على الإنترنت على سبيل المثال): بالنسبة للأساليب التي تؤثر على الإتاحة، فيمكن أن تتسب بتدمير أو محق الرسائل أو منع الوصول إلى الشبكة. في السرية، لدينا قراءة ومراقبة سير الرسائل من أو إلى الجهاز. وفيما يتعلق

2 الهدف من الهجمات يمكن أن يكون إتلاف أو تعطيل أو التجسس على المكونات التي لدينا، سواء كانت عتاداً أو برمجيات أو أنظمة اتصال.

بالصحة، فهو أي تعديل أو تأخير أو إعادة ترتيب أو تكرار أو تغيير للرسائل القادمة أو الخارجة.

1.1 التقنيات المستخدمة في الهجمات

الأساليب عديدة، ويمكن أن تعتمد على عنصر (عتاد أو برمجية) أو إصدار لعنصر معين. ولهذا، نحن بحاجة للحفاظ على البرمجيات محدثة للتصحيحات الأمنية التي تظهر وأن نتبع تعليمات المصنّع أو الموزِّع لحماية العنصر. رغم هذا، هناك دائماً تقنيات متبعة أو أساليب في أي وقت بعينه. فيما يلي بعض الملاحظات الموجزة المتعلقة بتقنيات الهجوم هذه الأيام:

- استغلالات العلل: أو استغلالات الأخطاء، أو "الاستغلالات" exploits فقط، سواء كانت لعتاد أو برمجية أو خدمة أو ميفاق أو نظام التشغيل نفسه (كالنواة مثلاً)، وبالعادة في إصدار معين منها. في الوضع الطبيعي، أي عنصر في الحاسوب معرض لأخطاء في تصميمه، أو ببساطة لأشياء لم يتم توقعها أو أخذها بعين الاعتبار. الثغرات يتم اكتشافها دورياً (وتعرف أحياناً بالاستغلالات أو العلل bugs)، والتي يمكن استغلالها لاختراق امن النظام. يتم عادة استخدام إما تقنيات عامة للاختراق، كتلك المشروحة فيما يلي، أو تقنيات معينة للعناصر المتأثرة. لكل عنصر متأثر من هو مسؤول عن إنشاء إصدارات جديدة أو تراقيع للتعامل مع هذه المشكلات - سواء كان المصنّع أو المطور او الموزع أو مجتمع جنو/لينكس -. وكمدراء، فمن مسؤوليتنا أن نبقى على اطلاع وأن نحافظ على سياسة مسؤولة للتحديثات لتجنب المخاطر المحتملة للهجمات. إذا لم تكن هناك حلول متاحة، فيمكننا أيضاً دراسة واستخدام بدائل للعنصر أو تعطيله إلى أن نجد حلًّا.

- الفايروس[3]: برنامج عادة يكون مدمجاً بغيره ويستخدم تقنيات للنسخ الذاتي والنقل. من الشائع دمج فيروسات في البرامج التطبيقية، ورسائل البريد الإلكتروني، أو دمجها في مستندات أو برامج تسمح بالتعامل مع وحدات برمجية "ماكرو"

3 لقد أشار الكاتب إلى هذا النوع من البرمجيات بكلمة "فايروس"، والأصح أن نطلق عليها "برمجيات خبيثة" Malware. وتشمل البرمجياتُ الخبيثةُ كلًّا من: الفيروسات، وأحصنة طروادة، وبرمجيات التجسس، والديدان، و rootkit، وغيرها ... ويمكن لهذه البرمجيات أن تستغل أي نقطة من نقاط الضعف الأمني المتوفرة للقيام بعمليات على النظام بسهولة، وفعل أي شيء، وتستخدم أكثر من تقنية للتخفي، والإندماج، والتنقل، إلخ ...

macro. هذه بالأحرى أعظم وباء أمني في هذه اللحظة.

أنظمة جنو/لينكس محمية تقريباً بشكل كامل من هذه الآليات لأسباب عديدة: في البرامج التنفيذية، لديها وصول محدود جدًّا إلى النظام، وبالتحديد إلى حساب المستخدم. وباستثناء المستخدم الجذر حيث يجب أن نكون حذرين جدًّا فيما يتم تنفيذه. لا يستخدم البريد ماكرو غير متحقق منها (عكس Outlook و Visual Basic Script في وندوز المعرضة لدخول الفيروسات عبرها)، أما في حالة المستندات، فنحن في حالة مشابهة ، حيث أنها لا تدعم لغات البرمجة النصية أو الماكرو غير المتحقق منه (مثل Visual Basic for Applications [والذي يعرف اختصاراً بالاسم VBA] في Microsoft Office).

وعلى أيّ حال، علينا أن ننتبه لما يمكن أن يحدث في المستقبل، حيث يمكن أن يتم عمل فيروسات مخصصة لجنو/لينكس باستغلال بعض العلل والثغرات. علينا أيضاً أن نلقي نظرة على أنظمة البريد حيث أننا حتى وإن لم نُنشئ فيروسات، فقد ننقلها، على سبيل المثال، إذا كان نظامنا يعمل كموجّه بريد، فيمكن أن تأتي رسائل محملة بفيروسات، ويمكن أن تُرسَل لآخرين بعد ذلك. يمكننا هنا أن نطبق سياسات الترشيح واكتشاف الفيروسات. وهناك آفة أخرى يمكن أن تدخل ضمن تصنيف الفيروسات، وهي رسائل السُخام spam، والتي رغم كونها لا تستخدم عادة كعناصر هجوم، فيمكن أن تعتبر مزعجة بسبب الانتشار الكبير الذي تظهر فيه، والتكاليف المالية التي يمكن أن تنجم عنها (في خسارة الوقت والموارد).

- الديدان: هي بالعادة نوع من البرامج التي تستغل وجود علة في النظام من أجل تنفيذ أكواد دون صلاحيات. ويتم استخدامها عادة لاستغلال موارد الجهاز، كاستخدام المعالج مثلاً، عندما تكتشف أن النظام لا يعمل أو أنه ليس قيد الاستخدام أو - بنيّة خبيثة - بهدف سرقة الموارد، أو استخدامها لإيقاف أو حجب النظام. تقنيات النقل والنسخ شائعة الاستخدام أيضاً.

- حصان طروادة (Trojan horse واختصاراً Trojan): برامج مفيدة تتضمن وظيفة معينة، ولكنها تخفي وظيفة أخرى تستخدم للحصول على معلومات من النظام أو من أجل تعريضه للاختراق. ومن الأمثلة على ذلك الاكواد ذات

النوع المحمول لتطبيقات الوب، مثل جافا وجافاسكربت وActiveX؛ وهذه عادة تطلب الموافقة ليتم تنفيذها (ActiveX في وندوز) أو يكون لديها نماذج محدودة لما يمكن فعله (جافا أو جافاسكربت). ولكن ككل البرمجيات، فلها أيضاً علل، وهي طريقة مثالية لنقل أحصنة طروادة.

- الأبواب الخلفية (back doors أو أبواب الأنفاخ trap doors): طرق للوصول إلى برنامج مخفي يمكن أن يُستخدَم لمنح وصول إلى النظام أو معالجة بيانات دون معرفتنا. ومن الآثار الأخرى المحتملة تغيير إعدادات النظام أو السماح بإدخال الفيروسات. الآلية المستخدمة يمكن أن تأتي مضمنة في بعض أنواع البرمجيات الشائعة أو كأحصنة طروادة.

- القنابل المنطقية logic bombs: برنامج مدمج في برنامج آخر يفحص حدوث ظروف معينة (مؤقتة، أو أفعال مستخدم، إلخ) لتفعيل نفسها وللقيام بأنشطة غير مصرح بها.

- Keyloggers: برنامج خاص متخصص في "اختطاف" التفاعل عبر لوحة مفاتيح وفأرة المستخدم. يمكن أن تكون برامج مستقلة، أو أحصنة طروادة مضمنة في برامج أخرى.

بالعادة، تحتاج هذه البرامج لأن يتم إدخالها في نظام مفتوح تم الوصول إليه (رغم أنها صارت تأتي مضمنة في أحصنة طروادة مثبتة أكثر فأكثر). تقوم فكرتها على اصطياد أيّ ضغطة مفاتيح، كأن يتم تسجيل كلمة المرور (لحسابات البنوك مثلاً) والتفاعل مع التطبيقات، والمواقع المُزارة، والنماذج المعبّأة، إلخ.

- الماسحات (port scanning): هي ليست هجوماً بالمعنى الحرفي، لكنها تمثل خطوة ابتدائية تتكون من جمع الأهداف المحتملين. وبشكل أساسي، تتكون من استخدام أدوات تسمح للشبكة بأن يتم فحصها من أجل إيجاد أجهزة بها منافذ مفتوحة – سواء كانت TCP أو UDP أو لموافيق أخرى – تشير إلى وجود خدمة معينة. على سبيل المثال، فحص الأجهزة للبحث عن المنفذ TCP 80 الذي يشير إلى وجود خادم وب، يمكننا من خلاله الحصول على معلومات عن الخادم والإصدار المستخدم من أجل استغلال الثغرات المعروفة فيه.

- المتنصّتات (sniffers): تسمح بالتقاط سير الحزم في الشبكة. يمكننا بالأدوات المناسبة أن نحلل سلوك الأجهزة: خوادم و عملاء، والموافيق المستخدمة، وفي العديد من الحالات الحصول على كلمات السرّ للخدمات غير الآمنة.

مبدئيًّا، لقد استُخدِمَت كثيراً لالتقاط كلمات مرور الخدمات غير الآمنة telnet, rsh, rcp, ftp، ... التي يفترض عدم استخدامها (استخدم الإصدارات الآمنة بدلاً من ذلك: ssh, scp, sftp). المتنصّتات (والماسحات) ليست بالضرورة أدوات هجوم، حيث يمكن أن تستخدم أيضاً لتحليل شبكاتنا واكتشاف الخلل، أو ببساطة لتحليل سير البيانات لدينا. في العادة يستخدم المخترقون تقنيات كل من الماسحات والمتنصتات للبحث عن ثغرات النظام، سواء لمعرفة البيانات لنظام غير معروف (الماسحات)، أو لتحليل تبادلاته الداخلية (المتنصّتات).

- الاختطاف Hijacking: وهي تقنيات تقوم على محاولة إبدال جهاز بحيث يستبدل أو يعيد إنتاج مهام خدمة في جهاز آخر تم اعتراض اتصاله. وهي شائعة في حالة نقل صفحات الوب و البريد الإلكتروني و الملفات. على سبيل المثال، في حالة الوب، يمكن أن يتم اختطاف جلسة، وسيكون بالإمكان إعادة إنتاج ما يقوم المستخدم بعمله، والصفحات المزارة، والرد على النماذج، إلخ.

- الفيض Buffer Overflow: تقنية معقدة نوعاً ما تستغل أخطاء البرمجة في التطبيقات. الفكرة الأساسية هي استغلال الطفح في ذاكرة التطبيق، سواء كانت طوابير أو مصفوفات أو غيرها. إذا لم يتم التحكم بالحدود، فيمكن أن يُنتج تطبيقٌ مهاجمٌ رسالةً أو بياناتٍ أكبر من المتوقع مما يتسبب بفشل. على سبيل المثال، العديد من تطبيقات C ذات التحكم السيّء بالذاكرة في المصفوفات، إذا تخطينا فيها الحد فيمكن أن تتسبب بإعادة كتابة كود التطبيق مما يتسبب بعطل أو انهيار في الخدمة أو الجهاز. إضافة إلى ذلك، يسمح أحد الأشكال الأكثر تعقيداً لأجزاء من البرنامج أن يتم استخدامها في الهجوم (برنامج C مصرّف، أو نص برمجي للصدفة)، مما يسمح بتنفيذ أيّ كود يرغب المهاجم بتنفيذه.

- حجب الخدمة (أو منع الخدمة Denial of Service – DoS): يتسبب هذا النوع من الهجوم بانهيار الجهاز أو عمل حمل زائد على خدمة أو أكثر، مما يجعلها غير قابلة للاستخدام. وهناك تقنية أخرى تدعى هجوم حجب الخدمة الموزّع (Distributed Denial of Service – DDoS)، والتي تعتمد على استخدام مجموعة من الأجهزة الموزعة من أجل إنتاج الهجوم أو الحمل الزائد على الخدمة. يتم حل هذا النوع من الهجوم عبر تحديث البرمجيات، حيث أنه في الوضع الطبيعي كل الخدمات التي لم تصمم لحمل معين تتأثر ولا يكون هناك تحكّم بالحمل. إن هجمات DoS و DDoS شائعة الاستخدام في الهجوم على المواقع أو خوادم DNS، والتي تتأثر بثغرات الخادم، كإصدارات بعينها من أباتشي و

BIND مثلاً. وهناك عامل آخر يستحق أن يأخذ بعين الاعتبار، وهو أنه يمكن أيضاً أن يتم استخدام جهازنا في هجمات من نوع DDoS عبر التحكم به من باب خلفي Backdoor أو حصان طروادة Trojan Horse.

من الأمثلة البسيطة لهذا الهجوم ما يعرف بهجوم SYN flood، والذي يحاول إنشاء حزم TCP تفتح اتصالاً، ولكن دون استخدامه في شيء، بل تركه مفتوحاً وحسب، هذا يستهلك موارد النظام على هيكليات بيانات النواة وموارد اتصالات الشبكة. إذا كان هذا الهجوم مكرراً مئات أو آلاف المرات، فيمكن أن تصير كل الموارد مشغولة دون أن تكون مستخدمة، وبهذا عندما يرغب المستخدمون باستخدام الخدمة يتم منعهم لأن الموارد مشغولة. وتعرف الحالة الأخرى بتفجير البريد الإلكتروني، أو ببساطة إعادة إرسال الرسائل (وعادة تكون معلومات المرسلة خاطئة) إلى أن يصير حساب البريد محملاً فوق طاقته، مما يتسبب بانهيار نظام البريد، أو جعله بطيئاً لدرجة لا يعود فيها النظام قابلاً للاستخدام. وإلى حد ما، فإن هذه الهجمات سهلة التنفيذ نوعاً ما إذا توفرت الأدوات المناسبة، وليس لها حل سهل، حيث أنها تستفيد من العمل الداخلي للموافيق والخدمات، نحتاج في هذه الحالات لاتخاذ إجراءات لاكتشاف هذه الخدمات ومن ثم السيطرة عليها.

- الخداع spoofing: تتضمن تقنيات الخداع أساليب عديدة (وعادة ما تكون معقدة جدًّا) لإعطاء معلومات خاطئة أو إظهار المستخدمين كمستخدمين آخرين في البيانات المنقولة (الأصل و/أو الهدف). من الأمثلة على الخداع:

  - إخفاء عنوان IP، وهي أظهار الجهاز وكأنه جهاز آخر، مما يسمح بإنشاء سير بيانات خاطئ أو اعتراض للبيانات التي تم توجيهها إلى جهاز آخر. وإذا ما اجتمعت مع هجمات أخرى، فيمكن أن تتخطى حتى جدار الحماية.

  - الخداع المتعلق بـ ARP، تقنيات معقدة (تستخدم DDoS)، تحاول إعطاء معلومات خاطئة عن عناوين المصدر والمتصلين على الشبكة، وذلك بمهاجمة معلومات ARP المختزنة في الأجهزة، بحيث يتم إبدال العناوين الحقيقية إلى أخرى في نقاط عديدة في الشبكة. يمكن لهذه التقنية أن تخرق كل أنواع الحماية – بما فيها جدران الحماية –، ولكنها تقنية بسيطة.

  - البريد الإلكتروني هو على الأرجح الأسهل بينها. ويعتمد على إنشاء رسائل بريد إلكتروني بمعلومات خاطئة، بما فيها

المحتوى وعنوان المصدر. لهذا النوع، من الشائع استخدام تقنيات تعرف بالهندسة الاجتماعية، وتقوم هذه التقنيات بشكل أساسي بخداع المستخدم بطريقة معقولة، ومن الأمثلة التقليدية الرسائل الخاطئة من مدير النظام، أو – على سبيل المثال – من المصرِف الذي فيه حسابنا الحالي، مشيرة إلى أنه كانت هناك مشاكل في الحسابات، وأنه يجب علينا أن نرسل معلومات سرية أو كلمة المرور السابقة من أجل حلها، أو طلب تغيير كلمة المرور إلى كلمة أخرى معينة. ومن الغريب أن هذه التقنية (والتي تعرف أيضاً بالتصيّد Phishing) مازالت تخدع عدداً كبيراً من المستخدمين. وحتى مع الأساليب البسيطة (من الهندسة الاجتماعية): فقد اعترف مخترق معروف بأن أسلوبه المفضّل كان عبر الهاتف. وكمثال، سنصف حالة لشركة تعطي شهادات (Verisign) والتي حصل لها المخترقون على توقيع البرمجيات الخاص بمايكروسوفت بمجرد الاتصال باسم شركة وإخبارهم بأن مشكلة ما قد حصلت وأنها تحتاج المفتاح مرة أخرى. والخلاصة أنه يمكن تخطي مراحل متقدمة من أمن الحاسوب بمكالمة هاتفية بسيطة أو عبر رسالة بريد إلكتروني فهمها المستخدم على منحنى خاطئ.

- الحقن SQL Injection: وهي تقنية موجهة نحو قواعد البيانات وخوادم الوب على وجه التحديد، والتي تستغل البرمجة الخاطئة لنماذج الوب، حيث لا تكون المعلومات المقدمة قد تم التحكم بها بشكل صحيح. والتي لا تتحقق من أن المعلومات المدخلة من النوع الصحيح (يجب أن تكون المدخلات مطابقة تماماً لما هو متوقع)، أو أن لا يكون هناك تحكم بالمحارف التي يسمح باستخدامها. تستغل هذه التقنية حقيقة كون المحارف التي يتم الحصول عليها من النموذج (كنموذج الوب مثلاً، ولكن يمكن أن يتم الهجوم عبر أي واجهة برمجة تطبيقات API تسمح بالوصول إلى قاعدة بيانات، كلغتي PHP و بيرل مثلاً) تُستخدم مباشرة للقيام بالمهام بلغة SQL، والتي ستهاجم قاعدة بيانات معينة (ليس هناك بالأصل وصول مباشر إليها). في العادة، إذا كانت هناك ثغرات وتحكم سيئ بالنماذج، فيمكن حقن أكواد SQL إلى نموذج بحيث تقوم هذه الأكواد بإنشاء استعلام SQL يقدم المعلومات المبحوث عنها. في الحالات المتقدمة، يمكن أن يتم الحصول على معلومات أمنية (كأسماء مستخدمي قاعدة البيانات وكلمات مرورهم)، أو حتى جداول قاعدة البيانات بأكملها، أو حتى التسبب بخسارة المعلومات أو الحذف الداخلي للبيانات. هذه التقنيات في بيئات الوب على وجه التحديد يمكن أن تكون خطرة، وذلك بسبب قوانين حماية

خصوصية المعلومات الشخصية والتي يمكن أن يهددها مستخدم من هذا النوع. في هذه الحالة، أكثر من كونها قضية متعلقة بأمن النظام نحن نتعامل مع مشكلة برمجية، والتحكم الدقيق بنوع البيانات المتوقع أن يتلقاها التطبيق، إضافة إلى التحكم الجيد بمعرفة الثغرات الموجودة في البرمجيات المستخدمة (قاعدة البيانات، وخادم الوب، وواجهة برمجة التطبيقات ...).

- البرمجة عبر المواقع Cross-site Scripting – XSS: مشكلة أخرى متعلقة ببيئة الوب، وعلى وجه التحديد في التعديل على أكواد HTML والنصوص البرمجية التي يمكن للمستخدم الحصول عليها عبر استعراض موقع معين يمكن أن يتم تغييره ديناميكيًّا. وبشكل عام يتم استغلال الأخطاء التي تظهر عندما يتعلق الأمر بالتحقق من كود HTML (كل المتصفحات لديها مشكلة مع هذه، وذلك بسبب تعريف HTML نفسه، والذي يسمح بقراءة أي كود HTML بغض النظر عن الأخطاء التي فيه). في بعض الحالات، يمكن أن يكون استخدام الثغرات مباشراً عبر النصوص البرمجية في صفحة الوب، ولكن عادة يكون للمتصفحات تحكم جيد بها. وفي نفس الوقت، فهناك بشكل غير مباشر تقنيات تسمح بإدراج نص برمجي – سواء عبر الوصول إلى كعكات المستخدم من المتصفح، أو عبر تغيير عملية التحويل من صفحة معينة إلى أخرى. هناك أيضاً تقنيات تعتمد على الإطارات، والتي يمكن أن تعيد توجيه أكواد HTML التي يتم عرضها، أو بتعليق المتصفح مباشرة. وعلى وجه التحديد، يمكن أن تكون محركات بحث المواقع عرضة لهذه الهجمات، وذلك بالسماح للنصوص البرمجية بأن يتم تنفيذها. وبشكل عام، هي هجمات ذات تقنيات معقدة، ولكنها مصممة لاصطياد معلومات، كالكعكات cookies، والتي يمكن أن تُستخدَم في جلسات، وبالتالي تسمح بأن يتم إبدال مستخدم معين بإعادة توجيه المواقع، أو بالحصول على معلوماته. ومرة أخرى من وجهة نظر النظام، فهذا يتعلق بالبرمجية المستخدمة. علينا أن نعلم بالثغرات المكتشفة في المتصفحات وأن نتحكم بها (والاستفادة قدر الإمكان من الموارد التي توفرها من أجل تجنب هذه التقنيات) والتحكم باستخدام البرمجيات (محركات البحث المستخدمة، وإصدارات خوادم الوب، وواجهات برمجة التطبيقات المستخدمة في تطوير البرمجيات).

وفيما يلي بعض النصائح الأساسية العامة المتعلقة بالأمن:

- التحكم بالعوامل المسببة للمشاكل: من أكثر العوامل التي يمكن أن تؤثر في الأمن هي سرية كلمات السر، والتي تتأثر بسلوك المستخدمين، هذا يسهل العمل في النظام نفسه من وجهة نظر المهاجم المحتمل. تأتي معظم الهجمات من داخل النظام، وبعبارة أخرى، بمجرد حصول المهاجم على وصول إلى النظام.

- المستخدمون، بما فيهم من ينسون كثيراً (أو الحمقى) الذين ينسون كلمات مرورهم بشكل مستمر، أو يذكرونها في محادثات، أو يتركونها مكتوبة على ورقة موضوعة أو ملصقة بجانب المكتب أو الحاسوب، أو من يقومون ببساطة بإعطائها لمستخدمين آخرين أو لبعض معارفهم. وهناك نوع آخر من المستخدمين، وهم من يكتبون كلمات سر يمكن التنبؤ بها، كأن تكون مطابقة لاسم المستخدم لديهم، أو رقم الهوية (أو الرقم الوطني)، أو اسم الزوجة أو الأم أو القطة ... إلخ.، والتي يمكن اكتشافها بسهولة بأقل قدر ممكن من المعلومات. وهناك حالة أخرى، وهي المستخدمون ذوي القدر الكافي من المعلومات، والذين لديهم كلمات مرور جيدة، ولكن علينا أن نبقي في بالنا أن هناك آليات قادرة على اكتشافها (كسر كلمات المرور، التنصّت، الخداع ...). علينا أن نبني "ثقافة" أمنية بين المستخدمين، وأن نجبرهم – عبر استخدام تقنيات – على تغيير كلمات مرورهم، دون استخدام كلمات اعتيادية، وأن يستخدموا كلمات مرور طويلة (أكثر من 6 أو 8 محارف)، إلخ. قامت مؤخراً العديد من الشركات والمؤسسات بتنفيذ تقنيات بجعل المستخدم يوقع على اتفاقية تجبر المستخدم على عدم إفشاء كلمة مروره أو القيام بأعمال التخريب أو الهجوم من حسابه (رغم أن هذا لا يمنع الآخرين من عمل هذا عبر المستخدم).

- عدم استخدام أو تشغيل برمجيات لا تستخدم آليات التحقق من التواقيع من أجل التحقق من أن حزمة البرمجيات مطابقة لما هو متوقع منها، كبصمات md5sum (الأمر md5sum) مثلاً، أو استخدام تواقيع GPG (الأمر gpg). يقدّم البائع أو الموزع توقيع md5 للفهم (أو صورة قُرصِهم) ويمكننا التحقق من صحتها. وتُستخدم هذه الأيام تواقيع لكل من الحزم المنفردة ومستودعات البرمجيات في التوزيعات كآليات للتحقق من إمكانية الاعتماد على المزود.

- لا تستخدم الحسابات ذات الصلاحيات الكبيرة (كالمستخدم الجذر) للاستخدام العادي للجهاز؛ لأن أي برنامج (أو

تطبيق) بهذه الطريقة سيكون بإمكانه الوصول إلى كل مكان في النظام.

- عدم الوصول عن بعد بالمستخدمين ذوي الصلاحيات أو تشغيل برامج يمكن أن يكون لديها صلاحيات. وخاصة إذا لم نكن نعرف أو إذا لم نكن قد اختبرنا مستويات أمن النظام.

- أن لا نستخدم عناصر لا نعرف كيف سلوكها، أو أن نحاول اكتشاف سلوكها عبر تنفيذها عدة مرات.

قد لا تكون هذه الإجراءات ذات جدوى، لكن إذا لم نكن قد حمينا النظام، فلن يكون لدينا تحكم بما يمكن أن يحدث، بل وحتى لا يمكن لأحد أن يضمن أنه لن يتمكن برنامج خبيث من التسلل وخرق الأمن إذا شغلناه بالصلاحيات المطلوبة. وبعبارة أخرى، نحتاج بشكل عام لأن نكون حذرين جداً لكل أنواع الأنشطة المتعلقة بالوصول وتنفيذ مهام ذات صلاحيات.

1.2 الإجراءات المضادّة

فيما يتعلق بما يمكن أن يُتخذ ضد هذه الأنواع من الهجوم التي يمكن أن تحدث، يمكننا أن نجد بعض الإجراءات الوقائيّة وبعض الإجراءات لاكتشاف ما يحدث في أنظمتنا. لننظر إلى بعض أنواع الإجراءات التي يمكن أن نتخذها في مجال منع الاختراق واكتشافه (سيتم ذكر أدوات مفيدة سنختبر بعضها لاحقاً):

- كسر كلمات السرّ: في الهجمات العنيفة brute force المصممة لكسر كلمات المرور، من الشائع تجربة الحصول على وصول عبر الولوج المتكرر؛ إذا تم الحصول على مدخلة، فسيتم الإخلال بأمن المستخدم، وسيكون الباب مفتوحاً لهجمات من أنواع أخرى، كهجمات الأبواب الخلفية، أو ببساطة تخريب حساب المستخدم. من أجل منع هذا النوع من الهجوم، نحتاج لتدعيم سياسة كلمات المرور، بالسؤال عن حد أدنى لطول كلمات المرور والتغيير الدوري لها. وهناك أمر علينا تجنبه، ألا وهو استخدام كلمات شائعة في كلمات المرور: معظم هذه الهجمات تتم عبر استخدام القوة العنيفة، بملف قاموس (يحوي كلمات في لغة المستخدم، ومصطلحات شائعة، ولهجة عامّيّة (دارجة)، إلخ). هذا النوع من كلمات المرور سيكون أول ما يتم كسره. سيكون من السهل أيضاً الحصول على معلومات من الضحية، كالاسم، ورقم الهوية (الرقم الوطني)، والعنوان، واستخدام هذه البيانات لاختبار كلمة المرور. لكل ما هو مذكور أعلاه، لا

ينصح بعمل كلمات مرور بها رقم الهوية أو الرقم الوطني أو الاسم (سواء اسم صاحب الحساب نفسه، أو اسم أحد أقاربه أو معارفه المقربين، إلخ)، أو العناوين، إلخ. ومن الخيارات الجيدة أن تكون كلمة المرور ما بين 6 و 8 محارف على الأقل، بمحتويات (أبجدية) أو رقمية، إضافة إلى المحارف الخاصة.

حتى وإن تم اختيار كلمة المرور بعناية، فقد لا تكون آمنة في الخدمات غير الآمنة. ولهذا، ينصح بتدعيم الخدمات باستخدا تقنيات التعمية التي تحمي كلمات المرور والرسائل. ومن ناحية أخرى، لمنع (أو عدم استخدام) أي خدمة لا تدعم التعمية، والتي تكون بالتالي عرضة للهجمات باستخدام أساليب مثل التنصّت؛ ومن بينها خدمات telnet, FTP, rsh, rlogin.

- استغلال العلل: تجنب وجود برامج غير مستخدمة، أو قديمة، أو غير حديثة (لأنها عفى عليها الزمن). طبّق أحدث التراقيع والتحديثات المتاحة لكل من التطبيقات ونظام التشغيل. افحص بالأدوات التي تكتشف الثغرات. ابق على اطلاع بالثغرات بمجرد اكتشافها [أولاً بأول].

- الفيروسات: استخدم برامج أو آليات مضادة للفيروسات، وأنظمة لترشيح الرسائل المشبوهة؛ تجنب تشغيل وحدات الماكرو (التي لا يمكن التحقق منها). علينا أن لا نكون محدودي الخيال فيما يتعلق بالإمكانيات المحتمل وجودها في الفيروسات، حيث أنها تتطور بشكل كبير يومياً، ومن الممكن تقنياً عمل فيروسات بسيطة يمكنها تعطيل الشبكات خلال دقائق (نحتاج فقط لأن ننظر إلى أحدث الفيروسات في عالم وندوز).

- الديدان: تحكم باستخدام الأجهزة والمستخدمين خارج أوقات الاستخدام المعتادة، وتحكم بسير البيانات الصادرة والواردة.

- أحصنة طروادة (Trojan): تفقد صحة البرامج دورياً باستخدام آليات التحقق sum و التواقيع الرقمية. اكتشاف سير البيانات المشبوهة الصادرة والواردة. استخدم جدران الحماية لإيقاف نقل البيانات المشبوهة. وهناك إصدار خطير من أحصنة طروادة وهو rootkit (المشروح أدناه)، والذي يقوم بأكثر من مهمة بفضل مجموعة متنوعة من الأدوات. من أجل التحقق من الصحة، يمكننا استخدام آليات التحقق مثل sum أو gpg، أو أدوات تجعل القيام بهذه العملية يتم

آلياً، مثل Tripwire و AIDE.

- الأبواب الخلفية: علينا الحصول على شهادات بأن البرامج لا تحوي أي باب خلفي مخفي غير موثّق من بائع أو مزود البرمجية، وبالتأكيد قبول البرمجيات من الأماكن التي توفر ضمانات فقط. عندما تكون البرمجية ملكاً لطرف ثالث، أو تأتي من مصادر يمكن أن تكون قد عدلت على البرمجية الأصلية، فسيضمّن بعض المصنّعين (والموزعين) إحدى آليات التحقق من البرمجية عبر أكواد sum والتواقيع الرقمية (من نوع md5 أو gpg). في أي وقت تكون فيه هذه الأمور متوفرة، فمن المفيد التحقق من البرمجيات باستخدامها قبل الاستمرار في التثبيت. يمكننا أيضاً أن نختبر النظام بشكل مكثف قبل تثبيته كنظام إنتاجي.

وهناك مشكلة أخرى، وهي التغيير المتأخر للبرمجيات. في هذه الحالة، يمكن أن تكون أنظمة التواقيع والتحقق مفيدة أيضاً، وذلك بإنشاء أكواد للبرمجيات المثبتة بالفعل، بحيث نتأكد من أنه لم يتم التعديل على البرمجيات الحيوية. أو النسخ الاحتياطية، والتي يمكننا مقارنتها من أجل اكتشاف أي تغييرات يمكن أن تكون قد تمت.

- القنابل المنطقية: هذا النوع يبقى مخبأً إلى أن يتم تفعيله في وقت معين أو عبر تفاعل من المستخدم. علينا التحقق من أنه لا يوجد وظائف غير تفاعلية في النظام في crontab أو type (بالأمر nohup type على سبيل المثال) ، والتي يتم تنفيذها دورياً، أو تنفذ في الخلفية لفترات طويلة. في هذه الحالة، يمكننا القيام بإجراءات وقائية وذلك بمنع الوظائف غير التفاعلية من المستخدمين، أو تفعيلها فقط للمستخدمين الذين يحتاجونها.

- مراقبات لوحة المفاتيح (keyloggers) والـ rootkits: تكون في هذه الحالة عملية وسيطة تحاول التقاط ضغطاتنا على لوحة المفاتيح ومحاولة تخزينها في مكان ما. سيكون علينا أن نتفقد ما إذا كانت هناك عملية قوية تظهر على أنها تعمل في حسابنا، أو تفقد إذا كان لدينا ملفاً مفتوحاً لا نعمل عليه في الوقت الحالي (على سبيل المثال، lsof يمكن أن يكون مفيداً، راجع دليل استخدامه عبر man)، أو اتصالات الشبكة، وذلك إذا كنا نتعامل مع مسجل ضغطات مفاتيح يدعم الإرسال إلى الخارج. لاختبار الوظائف الأساسية لمسجل ضغطات مفاتيح بسيط، فيمكننا أن نرى أمر النظام script (راجع man script). والآخر هو rootkit (والذي يتضمن بالعادة مسجلاً لضغطات المفاتيح أيضاً) هو في

العادة حزمة من عدة برامج بتقنيات عديدة، تسمح للمهاجم - بمجرد دخوله إلى الحساب - أن يستخدم العديد من العناصر، كمسجلات ضغطات المفاتيح، والأبواب الخلفية، وأحصنة طروادة (باستبدال أوامر النظام)، إلخ. من أجل الحصول على معلومات وأبواب للدخول إلى النظام، وعادة تكون مصحوبة ببرامج تمسح السجلات، من أجل التخلص مما يدل على حدوث الاختراق. والحالة الأكثر خطورة بينها هي rootkits التي تُستخدَم أو تأتي على شكل وحدات للنواة، مما يسمح لها بالعمل على مستوى النواة. من أجل اكتشافها، نحتاج للتأكد من أنه لا يوجد سير بيانات مع عنوان معين في الخارج. وهناك أداة مفيدة لاكتشافها وهي chrootkit.

- الماسحات: تعمل الماسحات على نظام تكراري واحد أو أكثر لفحص المنافذ المعروفة من أجل اكتشاف المفتوحة منها والخدمات التي تعمل (والحصول على معلومات عن إصدارات الخدمات) التي يمكن أن تكون عرضة للهجوم.

- المتنصتات: تجنب التنصت وبالتالي ستمنع إمكانية اعتراض اتصالك. من التقنيات المستخدمة البناء العتادي للشبكة، والتي يمكن أن تقسّم إلى أجزاء بحيث تمر البيانات في المنطقة التي ستستخدم فيها فقط، ووضع جدران حماية لوصل هذه الأجزاء، بحيث نكون قادرين على التحكم بالبيانات الصادرة والواردة. استخدم تقنيات التعمية بحيث لا يتمكن المتنصت من قراءة وتفسير هذه الرسائل. في حالة كل من المتنصتات والماسحات، فيمكننا استخدام أدوات مثل Wireshark و Snort لفحص شبكتنا، ولفحص المنافذ (بالماسحات) لدينا Nmap. يمكن اكتشاف المتنصتات على الشبكة بالبحث عن أجهزة في وضع إثرنت مشبوه (تستقبل أي حزمة تمر في الشبكة)؛ تلتقط بطاقة الشبكة في العادة البيانات التي تذهب إليها فقط (أو حزم broadcast و multicast).

- الاختطاف (أو السيطرة) Hijacking: طبق آليات لتعمية الخدمات، وطلب الاستيثاق، وإعادة الاستيثاق دورياً - إذا كان هذا ممكناً -. تحكم بالبيانات الصادرة والواردة باستخدام جدران الحماية. راقب الشبكة من أجل اكتشاف حركة البيانات المشبوهة.

- الطفح/الفيض Buffer Overflow: وهي معروفة كعلل أو ثغرات في النظام، ويتم حلها بتحديث البرمجيات. ويمكننا على أي حال من خلال السجلات أن نراقب الأحداث الغريبة، أو الانهيارات في الخدمات التي يفترض أن تكون

عاملة. يمكننا أيضاً زيادة التحكم بالعمليات والوصول إلى الموارد من أجل عزل المشكلة عند حدوثها في بيئات من الوصول المضبوط، كالذي يوفره SELinux (سنستعرضه لاحقاً في هذا الجزء).

- منع الخدمة (DoS) وغيرها، مثل SYN flood، و تفجير البريد: إجراءات حظر مرور البيانات غير الضرورية على شبكتنا (عبر استخدام جدران الحماية على كمثال). سيكون عليك التحكم بحجم الـ buffer، وعدد المستخدمين الذين ستستمع لطلباتهم، ووقت انتهاء الاتصال، وإمكانيات الخدمة، إلخ، في الخدمات التي تسمح لك بذلك.

- الخداع: أ) الخداع بتزييف عنوان IP. ب) خداع ARP. ج) البريد الإلكترونيّ. تحتاج هذه الحالات إلى تعمية قوية للخدمات، وتحكماً عبر استخدام جدران حماية، وآليات استيثاق معتمدة على عوامل عدّة (فمثلاً، أن لا تكون معتمدة على عنوان IP، حيث يمكن تخطيه)، وآليات يمكن تنفيذها تتحكم بالجلسات المُعتمدة على متغيرات عديدة للجهاز في نفس الوقت (نظام التشغيل، المعالج، عنوان IP، عنوان عتاد الشبكة، إلخ). ومراقبة أنظمة DNS أيضاً، والحفظ المؤقت لـ ARP، وتجمعات البريد، إلخ، من أجل اكتشاف التغييرات في المعلومات التي تناقض سابقاتها.

- الهندسة الاجتماعية: هذه ليست قضية تقنية بحقّ، لكن يجب علينا أن نتأكد من أن المستخدمين لا يجعلون الأمن أسوأ مما هو عليه. ويجب اتخاذ إجراءات مناسبة، كزيادة المعلومات أو تثقيف المستخدمين والتقنيين في مجال الأمن: التحكم في من يمكنه الوصول إلى المعلومات الأمنية الهامة، وفي أي حالة يمكنهم تمرير هذه الإمكانية إلى غيرهم. إن خدمات الصيانة والمساعدة في الشركة يمكن أن تكون نقطة حرجة: تحكم بمن لديه معلومات أمنية وكيف يستخدمها.

- فيما يتعلق بالمستخدمين النهائيين، فإن تحسين ثقافة كلمات المرور، وتجنب تركها مكتوبة في أي مكان يمكن لآخرين رؤيتها أو ببساطة إفشاؤها إلى الآخرين.

2 أمن النظام

للتصدي للهجمات المحتملة، يجب أن يكون لدينا آليات لمنعها واكتشافها ولاستعادة نظامنا بعدها.

للحماية المحلية، نحتاج لاختبار الآليات المختلفة للاستيثاق والصلاحيات للوصول إلى الموارد من أجل تعريفها بشكل صحيح، ولنكون قادرين على ضمان سرية وصحة المعلومات. في هذه الحالة، نحتاج لحماية أنفسنا من المهاجمين الذين حصلوا على وصول إلى نظامنا أو المستخدمين العدائيين الذين يرغبون بتخطي القيود المفروضة على النظام.

فيما يتعلق بأمن الشبكة، نحتاج لضمان أن الموارد التي نوفرها (إذا كنا نقدم خدمات معينة) لها المعاملات الضرورية من أجل السرية، وأنه لا يمكن أن يستخدم الخدمة أطراف خارجية غير مصرح لها، مما يعني أن الخطوة الأولى ستكون التحكم بأن الخدمات المقدمة هي ما نريد تقديمه فعلاً، وأننا لا نقدم خدمات أخرى لا يتم التحكم بها في نفس الوقت. في حالة الخدمات التي نكون عملاء لها، سيكون علينا التحقق من آليات الاستيثاق، بحيث يكون وصولنا إلى الخوادم الصحيحة، وأن لا تكون هناك حالات إبدال للخدمات أو الخوادم (والتي يكون اكتشافها في العادة صعباً).

وفيما يتعلق بالتطبيقات والخدمات نفسها، إضافة إلى ضمان الإعداد السليم لمستويات الوصول باستخدام صلاحيات واستيثاق لمستخدمين مخولين، ونحتاج لمراقبة إمكانية استغلال علل البرمجيات. إن أي تطبيق، مهما بلغ من الإتقان في تصميمه وتنفيذه، فلا بد وأن يحوي عدداً من الأخطاء – قلّت أم كثرت – التي يمكن استغلالها من أجل تخطي القيود المفروضة، باستخدام تقنيات معينة. في هذه الحالة، نفرض سياسة منع تتضمن إبقاء النظام محدثاً قدر الإمكان، بحيث نحدّث متى ما ظهر تحديث جديد، أو – إذا كنا محافظين – نبقي على الإصدارات الأكثر استقراراً فيما يخص الناحية الأمنية. ويعني هذا في العادة تفحص العديد من مواقع الأمن دورياً من أجل الاطلاع على أحدث المشاكل المكتشفة في البرمجيات والثغرات التي تنبع منها والتي يمكن أن تجعل أنظمتنا عرضة لمشاكل أمنية محلية أو عبر الشبكة.

3 الأمن المحلي

الأمن المحلي أساسي لحماية النظام، حيث أنها تلي المحاولةَ الأولى من الشبكة مباشرة، فهي الحاجز الثاني أمام الهجوم الذي يحصل على تحكم جزئي بالجهاز. إضافة إلى هذا، فإن معظم الهجمات تصل إلى استخدام الموارد الداخلية للجهاز.

3.1 محملات الإقلاع

فيما يتعلق بالأمن المحلي، يمكن أن تبدأ المشكلات من الإقلاع ومع الوصول الفيزيائيّ إلى الجهاز الذي يمكن أن يحصل المتسلل عليه. تبدأ إحدى المشكلات مع إقلاع النظام. إذا كان بالإمكان إقلاع النظام من قرص (فلاش أو CD ...)، فيمكن للمهاجم الوصول إلى بيانات أقراص جنو/لينكس (أو وندوز) بمجرد ضم نظام الملفات والانتقال إلى وضع المستخدم الجذر، دون الحاجة لأيّ كلمة مرور. في هذه الحالة، علينا حماية إقلاع النظام من BIOS – على سبيل المثال – بحماية الوصول عبر كلمة مرور، بحيث لا يسمح بالإقلاع من القرص الضوئي (الأقراص الحية مثلاً). من المنطقي أيضاً تحديث BIOS، حيث يمكن أن تكون بها مشاكل أمنية أيضاً. إضافة إلى ذلك، يجب أن نكون حذرين، لأن العديد من مصنعي العتاد يوفرون كلمة مرور إضافية معروفة (نوع من الأبواب الخلفية)، مما يعني أنه لا يمكننا الاعتماد حصرياً على هذا الإجراء.

المرحلة التالية هي حماية محمل الإقلاع – بغض النظر عن نوعه – بحيث لا يكون بإمكان المهاجم تعديل خيارات الإقلاع للنواة أو تعديل الإقلاع مباشرة (كما في حالة grub). يمكن لمحملات الإقلاع أن تُحمى بكلمة مرور.

في grub-legacy، يسأل الملف sbin/grub-md5-crypt/ عن كلمة المرور، وينشئ md5sum الناتج عنها. ثم يتم إدخال القيمة التي تم الحصول عليها إلى boot/grub/grub.conf/. تحت سطر timeout، نضيف:

```
password --md5 sum-md5-calculated
```

في Lilo، ننشئ إما كلمة مرور عامة:

```
password = <selected password>
```

أو للجزء الذي نريده:

```
image = /boot/vmlinuz-version
        password = <selected password>
        restricted
```

في هذه الحالة، تعني restricted أنه لن يكون بإمكاننا تغيير المعاملات التي يتم تمريرها إلى النواة من سطر الأوامر. علينا أن نكون حذرين بأن نجعل الملف etc/lilo.conf/ محمياً بحيث تكون صلاحيات القراءة والكتابة للجذر فقط (chmod 600).

وهناك أمر آخر متعلق بالإقلاع، وهو أنه في حال كان لدى أحد ما وصول إلى لوحة المفاتيح، فسيكون بإمكانه إعادة تشغيل النظام إذا ضغط CTRL+ALT+DELETE (معطل مبدئيًّا في بعض التوزيعات الحالية) فيتسبب بتوقف الجهاز. إن هذا السلوك مضبوط في etc/inittab/، بسطر شبيه بما يلي:

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

إذا تم وضع علامة التعليق على هذا السطر، فسيتم تعطيل هذه الإمكانية لإعادة التشغيل. ومن ناحية أخرى، يمكننا إنشاء ملف etc/shutdown.allow/ والذي يسمح لمستخدمين معينين بأن يعيدوا التشغيل.

3.2 الصدفة الآمنة Secure Shell – SSH

كانت كلمات المرور في أنظمة يونكس القديمة (والإصدارات الأولى من جنو/لينكس) معماة باستخدام خوارزميات DES (لكن بمفاتيح قصيرة، وبوجود استدعاء نظام مسؤول عن التعمية وإلغاء التعمية، وعلى وجه التحديد crypt؛ راجع دليل الاستخدام).

كانوا في الأصل في الملف etc/passwd/، في الحقل الثاني، وكمثال:

```
user:sndb565sadsd: …
```

لكن المشكلة تكمن في أن هذا الملف مقروء للجميع، مما يعني أنه يمكن لمهاجم الحصول على الملف واستخدام هجوم القوة العنيفة bruteforce إلى أن يتم إلغاء تعمية كلمات المرور التي يحويها الملف، أو استخدام هجوم القوة العنيفة مع قاموس.

الخطوة الأولى هي استخدام الملف shadow، والذي يتم حفظ كلمات المرور فيه هذه الأيام. يمكن فقط للمستخدم

الجذر قراءة هذا الملف، ولا أحد غيره. في هذه الحالة تظهر نجمة "*" في المكان الذي كانت فيه كلمة المرور قديماً. تستخدم توزيعات جنو/لينكس الحالية كلمات مرور من نوع shadow مبدئيًّا، ما لم يطلب منها عمل غير ذلك.

الخطوة الثانية هي تغيير نظام تعمية كلمات المرور إلى آخر أكثر تعقيداً وأصعب للكسر. توفر كل من فيدورا ودبيان كلمات سر بأنظمة تشفير قوية (مثل md5 و sha)، يسمح لنا عادة باختيار النظام عند التثبيت. علينا أن ننتبه مع كلمات السر المعماة باستخدام md5، لأننا إذا كنا نستخدم NIS، فيمكن أن تحدث مشكلة؛ عدا ذلك، سيستخدم كل العملاء والخوادم md5 لكلمات مرورهم. يمكن التعرف على كلمات السر في etc/shadow/، وذلك لأنها تبدأ بالرمز "1" (إشارة دولار ثم الرقم واحد ثم إشارة دولار).

وهناك أمور أخرى يمكن القيام بها، كإجبار المستخدمين على تغيير كلمات المرور دورياً (يمكن أن يكون الأمر change مفيداً)، وفرض قيود على حجم ومحتوى كلمات المرور، ومقارنتها بقواميس للكلمات الشائعة.

وفيما يتعلق بالأدوات، من المفيد الحصول على برنامج لكسر كلمات السر (أيْ برنامج للحصول على كلمات المرور)، من أجل فحص الوضع الأمني الحقيقي على حسابات مستخدمينا، وبالتالي نجبرهم على تغيير كلمات المرور التي اكتشفنا بأنها ضعيفة. من أكثر هذه الأدوات شيوعاً بين المدراء John the Ripper و crack. يمكن لهاتين الأداتين أيضاً أن تعملا مع قاموس، لذا سيكون من المفيد الحصول على قواميس ASCII باللغة الإنجليزية (يمكن إيجادها على الإنترنت). وهناك أداة أخرى اسمها slurpie يمكنها فحص عدّة أجهزة في نفس الوقت.

ومن القضايا التي ينبغي أخذها بعين الاعتبار تشغيل هذه الاختبارات على أنظمتنا. يجب علينا أن لا ننسى أن مدراء الأنظمة الأخرى (أو موفري خدمات الوصول والإنترنت) سيكون لديهم أنظمة مفعلة لاكتشاف الاختراق، وأنه يمكن أن نتّهم بمحاولة الاختراق، سواء من طرف الجهات المختصة (وحدات جرائم الحاسوب)، أو من طرف مزود خدمة الإنترنت لدينا بحيث يتم إيقاف اتصالنا. نحتاج لأن نكون حذرين جداً عند استخدام أدوات أمنية، حيث أنها دائماً تكون على الحد الفاصل بين الأمن والاختراق.

3.3 Suid & sticky bits

قضية هامة أخرى تؤثر على تصاريح خاصة تستخدم على الملفات والنصوص البرمجية.

إن sticky bits تستخدم بشكل خاص على المجلدات المؤقتة، حيث نرغب أن يكون باستطاعة بعض المجموعات (التي ليس لها قاسم مشترك أحياناً) لأي مستخدم أن يكون قادراً على الكتابة، وأن يكون باستطاعة مالك المجلد وحده حذف الملفات، أو مالك الملف الذي بداخل المجلد. وهناك مثال تقليدي على هذا الـ bit، وهو مجلد /tmp/. علينا أن نتأكد من عدم وجود مجلدات من هذا النوع، حيث يمكنها أن تسمح لأي أحد أن يكتب فيها، حيث يجب علينا أن نتأكد من أنه لا يوجد غير المجلدات الضرورية فقط، كالمجلدات المؤقتة. يتم وضع هذا الـ bit باستخدام الأمر (chmod +t) متبوعاً باسم المجلد، ويمكن إزالته باستخدام t-. وبالأمر ls سيظهر كمجلد بصلاحيات تشبه drwxrwxrwxt (لاحظ حرف t في نهايتها).

أما suid (اختصاراً لـ set uid) فيسمح لمستخدم بتنفيذه (ثنائياً كان أم نصاً برمجياً) بصلاحيات مستخدم آخر. يمكن أن يكون هذا مفيداً في بعض الحالات، ولكنه يمكن أن يكون خطيراً أيضاً. من الأمثلة على ذلك البرامج التي لديها suid للمستخدم الجذر: يمكن للمستخدم العادي – رغم عدم امتلاكه صلاحيات الجذر – أن يشغل برامج باستخدام setuid بحيث يمكن أن يكون بها صلاحيات المستخدم الجذر داخلياً. هذا خطر جداً في حالة النصوص البرمجية، حيث يمكن أن يتم تحريرها وتعديلها لعمل أي شيء. ولهذان يجب علينا أن نبقي هذه البرامج تحت السيطرة، وإذا لم يكن setuid ضرورياً، فسنحتاج لإزالته. يتم وضع هذا الخيار باستخدام chmod +s، سواء بتطبيقه على المالك (وعندها يسمى suid) أو على المجموعة (ويسمى في هذه الحالة sgid)؛ يمكن إزالته باستخدام s-. في حال عرضه باستخدام الأمر ls، فسيظهر الملف بصلاحيات مثل -rwSrw-rw (انتبه لحرف S) إذا كان suid، أما في حالة sgid فسيظهر بعد حرف w الثاني.

في حالة استخدام chmod بالأرقام الثُمانيّة، فسيتم استخدام أربعة أرقام، حيث ستكون الثلاثة الأخيرة للصلاحيات التقليدية rwxrwxrwx (تذكر أنه سيكون علينا إضافة الرقم 4 للقراءة r و 2 للكتابة w و 1 للتنفيذ x)، ورقم الأول له قيمة لكل تصريح خاص نرغب باستخدامه (يتم جمعها معاً): 4 (لـ suid)، و 2 (لـ sgid)، و 1 (لـ sticky).

3.4 تفعيل المضيفين

للنظام العديد من ملفات الإعداد الخاصة التي تجعل من الممكن تفعيل وصول عدد من المضيفين إلى إحدى خدمات الشبكة، ولكن أخطاء هذه الملفات ستجعل الأمن المحلي عرضة للهجوم لاحقاً. يمكننا أن نجد:

- ملفات rhosts. للمستخدمين: تسمح للمستخدم بتحديد عدد من الأجهزة (والمستخدمين) الذين يمكنهم استخدام حسابه عبر أوامر "r" (مثل rsh و rcp ...)، دون الحاجة لإدخال كلمة مرور الحساب. يحتمل أن يكون هذا خطيراً، حيث أن الإعداد الضعيف للمستخدم قد يسمح بدخول مستخدمين غير مرغوب بهم، أو قد تسمح لمهاجم (ذي وصول إلى حساب المستخدم) بأن يغير العنوان في rhosts. لكي يدخل بسهولة دون أي نوع من القيود. بالعادة علينا أن لا نسمح بإنشاء هذه الملفات، بل وعلينا أن نحذفها بشكل كامل وأن نعطل أوامر "r".

- etc/hosts.equiv/: هذه نفسها كما في ملفات rhosts. لكن على مستوى الجهاز، وتحدد أي خدمات وأي مستخدمين وأي مجموعات يمكنها استخدام أوامر "r" دون الحاجة لكلمة مرور. وأيضاً، يمكن لخطأ ما - كأن تضع إشارة الجمع + في سطر من ذلك الملف - أن يسمح بوصول **أيّ** جهاز. هذا الملف عادة لا يكون موجوداً هذه الأيام أيضاً، وهناك دائماً بديل ضمن ssh لأوامر r.

- etc/hosts.lpd/: كان يُستخدَم في نظام طباعة lpd لتحديد الجهاز الذي يسمح له بالوصول إلى نظام الطباعة. علينا أن نكون حذرين جدًّا، فإذا لم نكن نقدم خدمة، فعلينا أن نعطل الوصول إلى النظام، وإذا كنا نقدم خدمة، فعلينا أن نحصر الأمر في الأجهزة التي تستفيد منها بالفعل فقط. أو حاول التغيير إلى نظام CUPS أو LPRng، واللذان لديهما تحكم أكثر بكثير بالخدمات. إن نظام LPD هدف شائع لهجمات الديدان والطفح، وهناك العديد من العلل الهامة الموثقة. علينا أن نكون حذرين إذا كنا نستخدم هذا النظام وملف hosts.lpd.

3.5 وحدات PAM

إن وحدات PAM هي أسلوب يسمح للمدير بأن يتحكم بالكيفية التي تتم بها عملية استيثاق المستخدمين لتطبيقات معينة. يحتاج التطبيق لأن يكون قد تم إنشاؤه وربطه بمكتبات PAM. وبشكل أساسي، فإن ملفات PAM هي مجموعة من المكتبات

المشتركة التي يمكن تضمينها في التطبيقات كوسيلة للتحكم باستيثاق المستخدم فيها. ويمكن أيضاً تغيير أسلوب الاستيثاق (عن طريق إعداد وحدات PAM)، دون الحاجة لتغيير التطبيق.

وحدات PAM (المكتبات) موجودة في المجلد /lib/security/ (بهيئة كائنات ملفات يمكن تحميلها ديناميكياً). وتتواجد إعدادات PAM في المجلد /etc/pam.d/، حيث يظهر ملف إعدادات PAM لكل تطبيق يستخدم وحدات PAM. نجد إعدادات الاستيثاق للتطبيقات والخدمات مثل ssh، والولوج الرسومي عبر X Window System مثل xdm, gdm, kdm, xscreensaver, ... أو مثلاً عبر الولوج إلى النظام (الدخول باسم مستخدم وكلمة مرور). إصدارات "بام" القديمة استخدمت ملفاً (وهو في الوضع الاعتيادي /etc/pam.conf) لقراءة إعدادات "بام" منه إذا لم يكن المجلد /etc/pam.d/ موجوداً.

السطر المعتاد في هذه الملفات (في /etc/pam.d/) يحمل هذه الصيغة التالية (إذا كنا نستخدم /etc/pam.conf، فسيكون علينا إضافة الخدمة التي تنتمي إليها كحقل ابتدائيّ):

```
module-type control-flag module-path arguments
```

والتي تحدد:

1) نوع الوحدة module type: إذا كانت وحدة تتطلب استيثاقاً للمستخدم (auth)، أو لها وصول مقيّد (account)، أشياء علينا عملها عند ولوج أو خروج المستخدم (session)، وإذا كان على المستخدم أن يغير كلمة السرّ.

2) شارات التحكم control flags: وتحدد إذا كان الأمر مطلوباً required، أو أساسيًّا requisite، أو كافياً sufficient، أو إذا كان اختيارياً optional. هذا سياقها. وهناك واحد آخر أحدث يعمل في أزواج من القيم والأحداث.

3) مسار الوحدة.

4) معاملات يتم تمريرها إلى الوحدة (وتعتمد على كلّ وحدة بعينها). ولأن كل وحدة تحتاج للعديد من سطور الإعداد المشتركة، فيمكن أن يتم استخدام عمليات لتضمين تعاريف مشتركة لخدمات أخرى، وعلينا فقط أن نضيف سطراً

كالآتي:

```
@include service
```

وكمثال بسيط على استخدام وحدات PAM (في توزيعة دبيان)، استخدامها في عملية الولوج (لقد أشرنا أيضاً إلى السطور المضمنة في خدمات أخرى):

```
auth        requisite    pam_securetty.so
auth        requisite    pam_nologin.so
auth        required     pam_env.so
auth        required     pam_unix.so nullok
account     required     pam_unix.so
session     required     pam_unix.so
session     optional     pam_lastlog.so
session     optional     pam_motd.so
session     optional     pam_mail.so standard noenv
password    required     pam_unix.so nullok obscure min = 4 max = 8 md5
```

يحدد هذا وحدات بام الضرورية للتحكم باستيثاق المستخدمين أثناء الولوج. إحدى هذه الوحدات – وهي pam_unix.so – هي التي تتحقق من كلمة مرور المستخدم (بالنظر إلى الملفات المعنية، مثل passwd, shadow, ...).

والأخرى تتحكم بالجلسة، لرؤية متى كانت المدخلة الأخيرة، أو حفظ وقت دخول أو مغادرة المستخدم (بالأمر lastlog)، وهناك أيضاً وحدة مسؤولة عن التحقق مما إذا كان لدى المستخدم بريد لم يقرأه بعد (الاستيثاق مطلوب أيضاً) وأخرى تتحق من أن كلمة المرور تم تغييرها (إذا كان المستخدم مجبراً على عمل ذلك بعد أول ولوج) وأنها تحوي 4 إلى 8 حروف وأنه يمكن استخدان md5 للتعمية.

يمكننا في هذا المثال تحسين أمن المستخدم: يسمح الخياران auth و passwords لكلمة المرور أن تكون فارغة: هذا المعامل nullok للوحدة. سيسمح هذا بوجود مستخدمين دون كلمات مرور (وتكون بالتالي مصدراً محتملاً للهجمات). إذا أزلنا هذا الخيار، فلن نسمح بكلمات المرور الفارغة لعملية الولوج. يمكن عمل نفس الشيء مع ملف إعدادات كلمة المرور (في هذا المثال: أمر تغيير كلمة المرور)، والتي تقدم أيضاً nullok. وهناك إجراء آخر يمكن عمله، وهو زيادة الحدّ الأقصى لحجم كلمة المرور في كلي الملفين، وذلك باستخدام max = 16 مثلاً.

3.6 تغيير النظام

وهناك مشكلة أخرى يمكن أن تحدث، وهي استبدال الإعدادات أو الأوامر الأساسية للنظام، عبر وضع أحصنة طروادة أو أبواب خلفية، وذلك بتقديم برمجيات تستبدل أو تعدل سلوك برمجيات النظام بشكل طفيف.

وهناك حالة معتادة وهي إجبار المستخدم الجذر على تنفيذ أوامر نظام خاطئة، على سبيل المثال، إذا كان الجذر يستخدم النقطة "." في المتغير PATH، فسيسمح هذا للأوامر أن تُنفَّذ من مجلده، والتي قد تتسبب بإيجاد ملفات تستبدل أوامر النظام وتكون لها أولوية التنفيذ قبل أوامر النظام. يمكن عمل نفس الشيء مع المستخدم، ورغم أنها لن تؤثر على النظام بذلك القدر، وذلك بسبب محدودية صلاحيات المستخدم، إلا أنها ستؤثر على أمن المستخدم نفسه. وهناك حالة أخرى، وهي شاشات الولوج المزيفة، والتي تستبدل عملية الولوج الأصلية، وكلمة المرور، إلى برنامج خاطئ يخزن كلمة المرور المدخلة.

في حالة التغييرات من هذا النوع، فسيكون تنفيذ سياسة لمراقبة التغييرات أمراً مفيداً، سواء عبر حسابات التواقيع والمجاميع (gpg أو md5)، أو عبر نوع من برمجيات التحكم مثل Tripwire أو AIDE. أما بالنسبة لأحصنة طروادة فهناك عدة طرق لاكتشافها، أو استخدام أدوات مثل chrootkit إذا كانت تأتي من تثبيت rootkit معروفة.

4 نظام الأمن المحسّن – SELinux

بُني الأمن التقليدي في النظام على تقنيّة التحكم النسبيّ بالوصول Discretionary Access Control ويرمز له اختصاراً DAC، والتي يعني أن كل برنامج له تحكم كامل بالوصول إلى موارده (من ملفات أو ما شابه). إذا قرر برنامج ما (أو المستخدم ذو الصلاحيات) القيام بوصول خاطئ، سواء كان هذا بسبب إهمال أو خلل، فقد يؤدي هذا إلى ترك البيانات السرّيّة متاحة لغير المخولين. وبهذا يمكننا القول بأنه يمكن للمستخدم التحكم الكامل بممتلكاته (من ملفات أو غيره) والبرامج التي يشغّلها. عند تشغيل برنامج ما، فسيحمل نفس صلاحيات المستخدم الذي ينفّذه، ولهذا فسيعتمد أمن النظام على البرامج التي يتم تشغيلها والثغرات التي تحويها أو الأكواد الخبيثة التي من الممكن أن تكون مضمّنة بها، والتي يمكنها أن تؤثر على البرامج والموارد الأخرى التي يمكن للمستخدم الوصول إليها. وفي حالة المستخدم الجذر، سيعرّض هذا أمن النظام بأكمله للخطر.

كنظام بديل، تمكّن تقنيّة التحكم الإلزاميّ بالوصول Mandatory Access Control – MAC من تطوير سياسات أمنيّة يحدّدها مدير النظام بحيث تضبط أمكانيّة التحكم الكامل للنظام بحقوق الوصول لكلّ من الموارد المتاحة. فعلى سبيل المثال، يمكننا في نظام صلاحيات يونكس التقليدي إعطاء حق الوصول لملفات لنا صلاحيات عليها، لكن السياسات في نظام MAC تمكننا من التحكم أكثر وتحديد الملفات التي يمكن لعمليّة معيّنة الوصول إليها بدقّة، وأيّ مستوى من الوصول نريد. في MAC يتم إنشاء السياق الذي يحدد الأوضاع التي يمكن لموضوع فيها الوصول لموضوع آخر.[4]

أضيف إلى الفرع 2.6 (والأحدث) من النواة عنصر من نوع MAC وهو SELinux، والذي تضيفه الوزيعات باستمرار، ففيدورا وردهات يفعّلانه مبدئيًّا (لكن يمكن تعطيله أثناء أو بعد التثبيت)، وهو عنصر اختياريّ في دبيان.

يطبّق SELinux سياسات أمنيّة من نوع MAC والتي تسمح بصلاحيات وصول أكثر دقّة من نظام صلاحيات يونكس التقليديّ. يمكن مثلاً لمدير النظام أن يسمح بإضافة بيانات إلى ملف log معيّن، لكن لا يسمح بإعادة كتابة أو حذف أجزاء من هذا الملف (حيث يعد هذا النوع من التعديل من عادة المهاجمين لمسح آثارهم). وفي مثال آخر، يمكننا السماح لبرامج الشبكة الاتصال عبر المنفذ (أو المنافذ) التي تحتاجها، لكن نمنعها من الوصول لمنافذ أخرى، حيث يمكن أن تكون هذا تقنية

4 كلمة "موضوع" هي التي استخدمها المهندس صبري صالح KING SABRI في كتابه حول SELinux كترجمة لكلمة object في مثل هذا السياق.

يستخدمها مدير النظام لمساعدته في السيطرة على أحصنة طروادة أو الأبواب الخلفيّة.

قامت بتطوير SELinux وكالة الأمن القوميّ الأمريكيّة NSA مع مساهمات من شركات عديدة لأنظمة يونكس، والأنظمة الحرّة مثل Linux و BSD. لقد جُعِل SELinux حرًّا عام 2000، ومنذ ذلك الحين تم تضمينه في عدد من توزيعات GNU/Linux المختلفة.

لدينا في SELinux نموذج من نوع "نطاق"، حيث تعمل كلّ عمليّة داخل ما يسمّى سياقاً أمنيًّا، وكل مورد (ملف، مجلد، منفذ، ...إلخ) له نوع مرتبط به. هناك مجموعة من القواعد التي تحدد الأعمال التي يمكن تنفيذها على كلّ نوع في كلّ سياق. من فوائد هذا النموذج المعتمد على السياق أنّ السياسات المحدّدة يمكن تحليلها (هناك أدوات لهذا الأمر) لتحديد أيّ حركة للبيانات يسمح بها، وذلك لاكتشاف المسارات المختلفة للهجمات مثلاً، أو لتحديد ما إذا كانت السياسة الأمنيّة مكتملة بشكل جيّد لتغطية كل وصول محتمل أم لا.

في SELinux شيء يسمى SELinux policy database والتي تتحكم بكل نواحي SELinux، والتي تحدد السياقات التي يمكن لبرنامج استخدامها للعمل والتي تحدد أيضاً أنواع السياق التي يمكن الوصول إليها.

كل عملية نظام لها في SELinux سياق يتكون من ثلاثة أجزاء: هويّة ودور ونطاق. الهوية هي اسم حساب المستخدم، أو system_u لعمليات النظام، أو user_u إذا لم يكن للمستخدم سياسات محدّدة. يحدّد الدور السياسات المرتبطة. فمثلاً، لا يمكن لـ user_r أن يملك سياق sysadm_t (النطاق الرئيسيّ لمدير النظام). ولهذا لا يمكن لـ user_r بهويّة user_u الحصول على سياق sysadm_t في أيّ حال من الأحوال. السياق الأمني الذي يحدد دائماً بهذه القيم:

```
root:sysadm_r:sysadm_t
```

هو سياق لمدير النظام، ويحدد هويته ودوره والسياق الأمنيّ له.

على سبيل المثال، في جهاز به SELinux مفعّل (فيدورا في هذه الحالة)، يمكننا استخدام خيار Z- لـ ps لرؤية السياقات المرتبطة بالعمليات:

```
# ps -ax -Z
LABEL                              PID    TTY    STAT  TIME  COMMAND
system_u:system_r:init_t           1  ?   Ss     0:00  init
system_u:system_r:kernel_t         2  ?   S      0:00  [migration/0]
system_u:system_r:kernel_t         3  ?   S      0:00  [ksoftirqd/0]
system_u:system_r:kernel_t         4  ?   S      0:00  [watchdog/0]
system_u:system_r:kernel_t         5  ?   S      0:00  [migration/1]
system_u:system_r:kernel_t         6  ?   SN     0:00  [migration/1]
system_u:system_r:kernel_t         7  ?   S      0:00  [watchdog/1]
system_u:system_r:syslogd_t        2564   ?      Ss    0:00  syslogd -m 0
system_u:system_r:klogd_t          2567   ?      Ss    0:00  klogd -x
system_u:system_r:irqbalance_t     2579   ?      Ss    0:00  irqbalance
system_u:system_r:portmap_t        2608   ?      Ss    0:00  portmap
system_u:system_r:rpcd_t           2629   ?      Ss    0:00  rpc.statd
user_u:system_r:unconfined_t       4812   ?      Ss    0:00  /usr/libexec/gconfd-2 5
user_u:system_r:unconfined_t       4858   ?      Sl    0:00  gnome-terminal
user_u:system_r:unconfined_t       4861   ?      S     0:00  gnome-pty-helper
user_u:system_r:unconfined_t       4862   pts/0  Ss    0:00  bash
user_u:system_r:unconfined_t       4920   pts/0  S     0:00  gedit
system_u:system_r:rpcd_t           4984   ?      Ss    0:00  rpc.idmapd
system_u:system_r:gpm_t            5029   ?      Ss    0:00  gpm -m /dev/input/mice -t
exps2
user_u:system_r:unconfined_t       5184   pts/0  R+    0:00  ps ax -Z
user_u:system_r:unconfined_t       5185   pts/0  D+    0:00  Bash
```

وإذا استخدمنا خيار -Z في ls، فسنرى السياقات المرتبطة بالملفات والمجلدات:

```
# ls -Z
drwxr-xr-x   josep  josep  user_u:object_r:user_home_t    Desktop
drwxrwxr-x   josep  josep  user_u:object_r:user_home_t    proves
-rw-r--r--   josep  josep  user_u:object_r:user_home_t    yum.conf
```

ويمكننا من الطرفية معرفة سياقنا الحاليّ بتنفيذ:

```
$ id -Z
user_u:system_r:unconfined_t     5
```

بالنسبة لوضع العمل، يوفر SELinux وضعين يعرفان بالوضع المتساهل permissive والوضع التنفيذيّ enforcing.

5 ويمكن أن تكون أيضاً unconfined_u:unconfined_r:unconfined_t.

في الوضع المتساهل الوصول غير المصرح به مسموح ولكنّه مراقب، ويتم تسجيل أحداثه في التقرير (والذي يكون في العادة var/log/messages، وقد يكون في var/log/audit/audit.log في بعض التوزيعات). في الوضع المتشدّد لا يسمح بأيّ نوع من الوصول الذي تمّ منعه في السياسات التي تمّ ضبطها مسبقاً. يمكننا أيضاً إيقاف SELinux من ملف الإعدادات المتواجد عادة في etc/selinux/config/، وذلك بجعل قيمة SELINUX=disabled.

علينا الانتباه عند تفعيل وتعطيل SELinux، خاصّة مع عناوين السياقات في الملفات، حيث يمكن أن تُفقد هذه العناوين أو أن لا تُنشأ أثناء التفعيل أو التعطيل. علينا الانتباه أيضاً عند عمل نسخ احتياطيّة عن نظام الملفات، فعلينا التأكد من أن عناوين SELinux موجودة في النسخة الاحتياطيّة.

هناك مشكلة أخرى علينا أخذها بعين الاعتبار وهي العدد الكبير لقواعد السياسة الأمنية التي يمكن أن تسبب تقييداً في التحكّم في الخدمات. في مواجهة نوع معيّن من الأعطال، من المهم أن نتأكد أولاً إذا ما كان SELinux يمنع العمل بشكل سليم بسبب سياسة أمنية متشدّدة أكثر من اللازم (اطّلع على القسم المتعلّق بانتقادات SELinux) أو خيارات لم نتوقّع أنّنا فعّلناها (قد يحتاج إلى تغيير إعدادات المتغيرات المنطقيّة كما سنرى لاحقاً).

يدعم SELinux نوعين مختلفين من السياسات التي يتمّ تنفيذها، وهي: موجّهة targeted ومتشدّدة strict. في السياسة من النوع الموجّه، معظم العمليات تعمل دون قيود، إلا أن هناك عمليّات معيّنة فقط (كبعض المراقبات) توضع في سياقات أمنيّة مختلفة محددة بسياق أمنيّ. في السياسة من النوع المتشدّد، كل العمليات مرتبطة بسياق أمنيّ ومحدّدة بسياسات معيّنة، وبذلك فإنّ أيّ عمل يتم التحكّم به بالسياسات المضبوطة. مبدئيًّا، هذان هما نوعا السياسات المحدّدة بشكل عام، إلّا أنّ المواصفات تسمح بإضافة المزيد.

وهناك حالة خاصة للسياسات، وهي الأمن متعدد المستويات (MLS)، وهي سياسة متعددة المستويات من النوع المتشدّد. وتقوم فكرتها على تعريف عدة مستويات أمنية ضمن نفس السياسة، وبحيث يكون في السياقات الأمنية حقل إضافيّ يحوي مستوى الوصول المرتبط بها. هذا النوع من السياسات الأمنية (مثل MLS) يستخدم في المؤسسات الحكومية والعسكرية، حيث هناك بنية هرمية بمستويات مختلفة من المعلومات المخولة، ومستويات للوصول العام، وإمكانيات مختلفة للأنشطة في كل

مستوى. من أجل الحصول على شهادات أمنية، يجب أن يكون لدينا هذا النوع من السياسة الأمنية.

يمكننا أن نحدد أي نوع من السياسات الأمنية سيتم استخدامه في etc/selinux/config/، المتغير SELINUXTYPE. في الوضع الطبيعي سيتم تثبيت السياسة ذات العلاقة وإعداداتها في المجلدات / etc/selinux/SELINUXTYPE/ – على سبيل المثال –، ويمكننا أن نجد في المجلد الفرعي للسياسة الأمنية الملف الثنائي للسياسة الأمنية مصرّفاً (وهو ما يتم تحميله في النواة عندما يتم تشغيل SELinux).

4.1 المعمارية

تتكون معمارية SELinux من العناصر التالية:

- أكواد على مستوى النواة.
- مكتبة Selinux مشتركة.
- السياسة الأمنية (قاعدة البيانات).
- أدوات.

لنلقِ نظرة عل بعض الأمور التي علينا أخذها بعين الاعتبار فيما يتعلق بكل عنصر:

- تراقب أكواد النواة نشاطات النظام وتتأكد من أن العمليات المطلوبة مسموح بها في ظل إعدادات السياسة الأمنية الحالية لـ SELinux، وعدم السماح للعمليات غير المصرح بها، وبالعادة تنشئ مدخلات في التقارير حول العمليات التي تم منعها. هذه الأكواد مضمنة حالياً في الأنوية منذ 2.6، وهي متوفرة للأنوية الأقدم على شكل تراقيع.

- معظم أدوات وعناصر SELinux غير المرتبطة مباشرة باستخدام النواة تستخدم مكتبة مشتركة تُدعى libselinux1.so، والتي توفر واجهة برمجة تطبيقات للتفاعل مع SELinux.

- إن السياسة الأمنية هي ما يتم تضمينه في قاعدة بيانات قواعد SELinux. عندما يبدأ النظام العمل (ويكون SELinux مفعلاً)، يحمل ملف السياسة الثنائيّ، والذي يكون في العادة موجوداً في /etc/security/selinux (ولكن يمكن أن يختلف هذا المكان اعتماداً على التوزيعة).

يتم إنشاء ملف السياسة الثنائي كنتيجة لتصريف الملفات المصدرية للسياسة الأمنية وبعض ملفات الإعداد (عبر make). بعض التوزيعات (كفيدورا) لا تثبت المصادر بشكل مبدئيّ، والتي يمكننا إيجادها عادةً في /etc/security/selinux/src/policy أو في /etc/selinux. وعادة ما تتكون هذه المصادر من مجموعات عديدة من المعلومات:

- الملفات المتعلقة بالتصريف، وهي makefile والنصوص البرمجية المصاحبة له.
- ملفات الإعداد الابتدائيّة، والأدوار والمستخدمين المرتبطين.
- ملفات تنفيذ النوع، والتي تحوي معظم جُمَل لغة السياسة المرتبطة بسياق معين. علينا أن نأخذ بعين الاعتبار أن هذه الملفات كثيرة، وعادة ما تتكون من عشرات الآلاف من السطور، مما يعني أننا قد نواجه مشاكل في إيجاد العلل أو تحديد التغييرات في السياسة الأمنية.
- والملفات التي تخدم في عنونة سياقات الملفات والمجلدات أثناء التحميل أو في لحظات معينة.

- الأدوات: تتضمن أوامر تستخدم لإدارة واستخدام SELinux. ونسخاً معدلة من أوامر لينكس المعيارية. وأدوات لتحليل السياسات الأمنية وللتطوير.

لننظر من هذا القسم الأخير على الأدوات الأساسية التي يمكننا إيجادها عادة:

من الأوامر الأساسية:

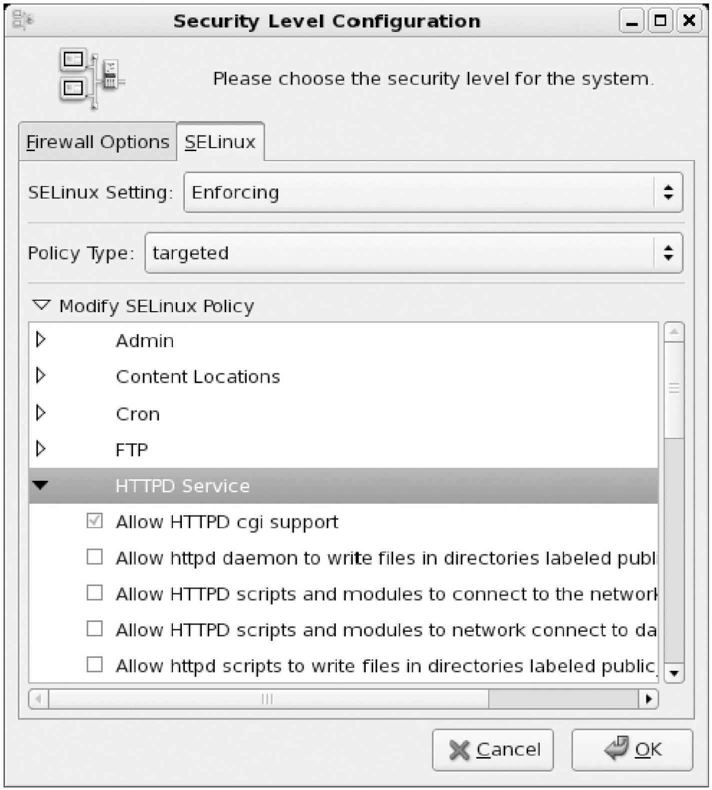
الاسم	الاستخدام
chcon	تَسِم ملفاً معيناً، أو مجموعة من الملفات، بسياق معين.
checkpolicy	يقوم بالعديد من الأنشطة المتعلقة بالسياسات، بما فيها تصريف السياسات الأمنية إلى ملفات ثنائيّة، والتي

	تتم عادة عبر عمليات ملف makefile.
getenforce	تنشئ رسالة بوضع SELinux (سواء permissive أو enforcing)، أو deactivated إذا كان معطلاً.
getsebool	تجلب قائمة المتغيرات الثنائية، وبعبارة أخرى، قائمة الخيارات المفعلة والمعطلة لكل سياق مرتبط بخدمة أو خيار عام للنظام.
newrole	تسمح بنقل مستخدم من دور إلى آخر.
runn_init	تستخدم من أجل تفعيل خدمة (start, stop)، للتأكد من أنها تتم ضمن نفس السياق الذي تعمل به تلقائيًا عندما تقلع (عبر init).
setenforce	يغير وضع selinux، الرقم 0 يعني متساهل permissive، و 1 يعني تنفيذيّ enfocing.
setfiles	تعنون المجلدات والمجلدات الفرعية بالسياقات المناسبة، وتستخدم في إعداد SELinux الابتدائيّ.
setstatus	تحصل على حالة النظام عبر SELinux.

وهناك برامج أخرى شائعة أيضاً تم تعديلها لتدعم SELinux، مثل:

- cron: تم تعديله ليتضمن سياقات الوظائف التي يتم تنفيذها عبر cron.
- Login: تم تعديله لإنشاء السياق الأمني الابتدائي للمستخدمين عندما يلجون في النظام.
- Logrotate: تم تعديله ليحافظ على سياق التقارير عندما يتم تصريفها.
- Pam: تم تعديله لينشئ السياق الابتدائي للمستخدم ولاستخدام واجهة برمجة تطبيقات SELinux للحصول على وصول مخول إلى معلومات كلمات المرور.
- Ssh: تم تعديله لإنشاء السياق الابتدائي عندما يلج المستخدم في النظام.
- والعديد من البرامج الإضافية التي تعدّل /etc/passwd و /etc/shadow.

وأيضاً، بعض التوزيعات تتضمن أدوات لإدارة SELinux، مثل أدوات الواجهات الرسومية التي تتضمن أدوات عديدة لإدارة وتحليل السياسات. إضافة إلى أدوات محددة للتحكم بالسياقات المرتبطة بالخدمات المختلفة التي يدعمها SELinux في التوزيعة، فمثلاً الأداة ذات المستوى في فيدورا system-config-security بها قسم لضبط SELinux يمكننا أن نراه في الشكل التالي:



شكل 1: واجهة في فيدورا لضبط SELinux

يمكننا أن نرى في هذا الشكل الإعدادات الثنائية للخدمات المختلفة وخيارات عامة، بما فيها خادم الوب. يمكننا أيضاً أن نحصل على هذه القائمة من الأمر getsebool، ويمكننا – بالأمر setsebool أو togglesebool – أن نفعل أو نعطل الخيار.

في فيدورا على سبيل المثال، نجد دعم المتغيرات الثنائية (وغيره) لكل مما يلي: cron, FTP, http (apache), DNS, grub, lilo, nfs, nis, cups, pam, ppd, samba، والحماية ضد الوصول غير المصرح به إلى ذاكرة العمليات، إلخ.

يسمح ضبط المتغيرات الثنائية لسياسة SELinux أن تُطوَّع أثناء التشغيل. ويتم استخدام هذه المتغيرات كقيم ظرفية لقواعد السياسة المطبقة، والتي تسمح بتغيير السياسة دون الاضطرار لتحميل واحدة جديدة.

4.2 النقد

انتقد بعض المدراء وخبراء الأمن SELinux، وذلك لكونه معقداً جداً للضبط والإدارة على وجه التحديد. ويدعي هؤلاء بأنه بسبب التعقيد الداخلي له، فحتى المستخدمون ذوو الخبرة يمكن أن يقعوا في أخطاء، مما يجعل ضبط SELinux غيرَ آمن وغيرَ قابل للاستخدام والنظامَ ضعيفاً. ورغم أن هذه قضية جدلية إلى حد ما، فحتى مع وجود SELinux مضبوط بشكل

سيء، فستظل صلاحيات UNIX سارية، فلن يسمح SELinux بالقيام بعملية لم تسمح بها الصلاحيات الأصلية، وفي الحقيقة، يمكننا أن ننظر إلى SELinux إلى طبقة أمنية إضافية أكثر تشدّداً.

قد تتأثر أيضاً عوامل الأداء، وذلك بسبب الأحجام الهائلة للسياسات، والتي تستخدم كمّاً كبيراً من الذاكرة وتأخذ كثيراً من الوقت في التحميل، وأيضاً - في بعض الحالات - بسبب معالجة القواعد. نحتاج لأن نبقي في بالنا أننا نتعامل مع نظام ذي عشرة آلاف قاعدة سياسة أمنية عملياً. وأنه يمكن أن يكون هذا الرقم حتى أكبر من هذا إذا اخترنا النوع المتشدد strict من السياسة الأمنية والذي نحتاج معه لتحديد كل الخيارات التي سيتم التحكم بها. وفي الوضع الاعتيادي، تسمح معالجة السياسة بالهيئة الثنائية واستخدام المتغيرات الثنائية لتعطيل القواعد للنظام بأن يُستخدَم بكفاءة أكبر.

ومن النواحي التي تزعج المدراء هي المشكلة الإضافية في تحديد السبب الأساسيّ وراء الخلل في حال حدوثه. ولأنه من الشائع لنا أن نجد في النهاية أن المشكلة قد نبعت من إعداد SELinux المتشدّد بشكل كبير جدًّا (وذلك ربما بسبب عدم الوعي من طرف المستخدم) لخدمة بعينها.

في النهاية، علينا أن نشير إلى الدعم الممتد الذي يقدمه SELinux في مجال الأمن، وإلى أننا كمدراء نحتاج لأن نكون واعين للإمكانات والمخاطر لأي تقنية جديدة نتبعها.

5 أمن الشبكة

5.1 عميل الخدمة

كعملاء لخدمة ما، نحتاج بشكل أساسي لأن نتأكد من أننا لا نضع مستخدمينا في منطقة الخطر (أو يضعوا هم أنفسهم فيها) باستخدام خدمات غير آمنة. تجنب استخدام الخدمات التي لا تستخدم تعمية البيانات وكلمات المرور (مثل FTP و Telnet و البريد غير الآمن). استخدم تقنيات الاتصال المعمّى، مثل SSH و SSL.

وهناك نطقة أخرى مهمة تتعلق بتقنيات الإبدال المحتمل لخوادم إلى أخرى خاطئة أو اختطاف الجلسات. سنحتاج في هذه الحالات لأن تكون لدينا آليّة قوية للاستيثاق تسمح لنا بالتحقق من مصداقية الخوادم (في SSH و SSL بعض هذه الآليات مثلاً). وسيكون علينا أيضاً أن نتحقق من الشبكة بالبحث عن الدخلاء الذين يحاولون وضع خوادم، إضافة إلى تطبيق خدمات ترشيح صحيحة للحزم باستخدام جدران الحماية – التي تسمح لنا بإزالة حزمنا من طلب ما واستخدام الخوادم الصحيحة – بالتحكم بالحزم القادمة التي نستقبلها كردّ.

5.1 الخادم: inetd و xinetd

كما رأينا، يتم ضبط الخدمات من أماكن مختلفة:

- في /etc/inetd.conf أو المجلد المكافئ لها في /etc/xinetd.d/: هذه الخوادم نوع من الخوادم الخارقة superservers، حيث تتحكم بالخدمات الثانوية وظروف تشغيلها. إن خدمة xinetd مستخدمة في كل من دبيان وفيدورا (ويمكن استخدام inetd أيضاً كخيار بديل).

- الخوادم التي يتم تشغيلها أثناء الإقلاع: سيكون لدينا مجموعة من الخدمات التي يتم بدؤها عند التشغيل اعتماداً على مستوى الإقلاع. ينشأ بدء التشغيل في المجلد المرتبط بمستوى الإقلاع. فعلى سبيل المثال، مستوى الإقلاع المبدئي هو الثاني، ويتم بدء تشغيل الخدمات من المجلد /etc/rc2.d/، ويتم ذلك عبر روابط بالنص البرمجي مضمنة في /etc/init.d/، والتي تعمل بالمعامل start أو stop أو restart حسب الحالة.

- خدمات أخرى من نوع RPC: وهي مرتبطة باستخدام استدعاءات بعيدة بين الأجهزة، مثل NIS و NFS. ويمكننا أن نعرف أيها من هذا النوع بالأمر rpcinfo -p.

تتضمن ملفات الدعم الأخرى (بمعلومات مفيدة): etc/services/ الذي يتكون من قائمة بكل الخدمات المحلية وخدمات الشبكة المعروفة باسم الميفاق – مثل TCP أو UDP أو غيره – المستخدم للخدمة والمنفذ الذي يستخدمه؛ etc/protocols/ قائمة بالموافيق المعروفة؛ و etc/rpc/ قائمة بخوادم RPC مع المنافذ المستخدمة. تأتي هذه الملفات مع التوزيعة، وهي نوع من قواعد البيانات التي تستخدمها أوامر وأدوات الشبكة. وعلينا أن نذكر أنها ملفات تاريخية، والتي لا تتضمن بالضرورة تعريفاً بكل الموافيق والخدمات؛ ويمكننا كذلك أن نبحث في القوائم المختلفة على الإنترنت لمنافذ معروفة[6].

من المهام الأولى للمدير تعطيل كل الخدمات التي لا يتم استخدامها، والقراءة عن استخدام الخدمات والبرمجيات التي تستخدمها.

في حالة etc/inetd.conf/، فعلينا فقط أن نجعل سطر الخدمة التي يفترض أن يتم تعطيلها تعليقاً، وذلك بوضع الرمز # في بداية السطر.

في نموذج xinetd للخدمات، والمستخدم في فيدورا ودبيان، يقبع الإعداد في الملف etc/xinetd.conf/، حيث يتم تحديد القيم المبدئية للتحكم بالتقارير، ويتم ضبط كل من الخدمات الثانوية عبر ملف في المجلد etc/initd.d/. ويتم في كل ملف تحديد معلومات الخدمة، وبشكل يكافئ ما يظهر في inted.conf، و في هذه الحالة، يكفي لتعطيل خدمة إدخال السطر disable=yes ضمن ملف الخدمة. لـ xinetd إعداد أكثر مرونة من inetd، حيث تفصل بين إعدادات الخدمات المختلفة في ملفات منفصلة، وبها عدد لا بأس به من الخيارات لتحديد الاتصالات إلى خدمة، بعددها وإمكانياتها؛ ويسمح كل هذا بتحكم أفضل بالخدمة، ويمكننا بالإعداد المناسب تجنب بعض الهجمات التي تقوم على منع أو حجب الخدمة (DoS أو DDoS).

وفيما يتعلق بخدمات مستوى التشغيل من أوامر التوزيعة، فقد ذكرنا بالفعل عدداً من الأدوات التي تسمح للخدمات بأن يتم تفعيلها أو تعطيلها في الوحدة المتعلقة بالإدارة المحلية. وهناك أيضاً أدوات رسومية مثل ksysv من KDE، و -system

6 وهنا واحدة: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

config-services و ntsysv من فيدورا (وفي دبيان ننصح بكل من sysv-rc-conf و rcconf و bum). وفي مستوىً أدنى، يمكننا الذهاب إلى مستوى التشغيل الذي نرغب به (/etc/rcX.d/ وتعطيل الخدمات التي لا نرغب بها، وذلك بتغيير الحرف الذي يبدأ به اسم الملف من S أو K إلى أي نصّ آخر: فعلى سبيل المثال، من الممكن تغيير S20ssh إلى STOP_S20ssh، ولن تعمل بعد ذلك؛ وفي المرة القادمة التي نحتاجها فيها، يمكننا إزالة ما أضفناه إلى البداية وسيتم تفعيلها مجدداً. أو ربما الاستخدام المستحسن لأدوات بسيطة لوضع أو إزالة أو تفعيل خدمة معينة (مثل service و chkconfig في فيدورا، أو ما يقابلها في دبيان، مثل update-rc.d و invoke-rc.d).

وهناك ناحية أخرى، وهي تعطي الخدمات غير الآمنة. في أنظمة يونكس قديماً، كانت تستخدم أنظمة لنقل الملفات مثل FTP مع اتصالات بعيدة مثل telnet، وأوامر تعمل عن بعد (login او copy)، والتي يبدأ كثير منها بالحرف r (مثل rsh, rcp, rexec ...). ومن المخاطر الأخرى المحتملة خدمات finger و rwhod، التي كانت تسمح بالحصول على المعلومات من شبكة مستخدمي الأجهزة؛ ويكمن الخطر هنا من المعلومات التي يمكن أن يحصل المهاجم عليها والتي ستجعل عملية الهجوم أسهل. يجب عدم استخدام أي من هذه الخدمات حالياً بسبب الخطر المحتمل الذي يتبعها. وفيما يتعلق بالمجموعة الأولى:

1) في النقل عبر الشبكة، لا تعمي telnet و ftp كلمات المرور، ويمكن لأيّ كان الحصول على كلمات سر الخدمة أو الحسابات المرتبطة بها (باستخدام متنصتات sniffers مثلاً).

2) وفي كل من rsh و rcp و rexec مشكلة وهي أنه – في ظل ظروف معينة – لا تكون كلمات السرّ ضرورية حتى (على سبيل المثال، إذا تم تشغيلها من أماكن مصرح لها في الملف rhosts.)، مما يعني – ونقولها مرة أخرى – أنها غير آمنة، وتترك الباب مفتوحاً على مصراعيه للهجمات.

والبديل هو استخدام عملاء وخدمات آمنة تدعم تعمية الرسائل والاستيثاق للمتصلين. هناك بدائل آمنة للخوادم التقليدية، ولكن الحل الاكثر شيوعاً هذه الأيام هو حزمة OpenSSH (والتي يمكن أن تُدمج أيضاً بـ SSL لبيئات الوِب). يوفر OpenSSH حلول معتمدة على الأوامر ssh و scp و sftp مما يسمح باستبدال العملاء والخوادم القدامى (باستخدام مراقب يُدعى sshd). يسمح الأمر ssh بالقيام بالوظائف القديمة لخدمة telnet و rlogin و rsh وغيرها، ويمكن أن نعتبر scp مكافئاً

لـ rcp، و sftp مكافئاً لـ ftp.

وفيما يتعلق بـ ssh، علينا أن نتأكد أيضاً أننا نستخدم الإصدار الثاني من ssh. للإصدار الأول بعض الثغرات المعروفة؛ علينا أن نكون منتبهين عندما نثبت OpenSSH، وأذا لم نكن محتاجين للإصدار الأول، أن نثبت دعم الإصدار الثاني فقط (انظر إلى خيارات الميفاق في ملف الإعداد /etc/ssh/ssh_config).

إضافة إلى ذلك، فمعظم الخدمات التي نتركها مفعلة في أجهزتنا يجب أن يتم ترشيحها فيما بعد عبر جدار حماية للتأكد من أنها لا تُستخدم أو تُهاجم من طرف أناس لم تكن موجهة لهم.

6 اكتشاف الاختراق

إن الهدف من أنظمة اكتشاف الاختراق هي التقدم خطوة للأمام. فبمجرد أن نكون قادرين على ضبط أمننا بشكل سليم، فستكون الخطوة التالية هي اكتشاف ومنع الاختراق بشكل فعّال.

تُنشئ أنظمة اكتشاف الاختراق Intrusion Detection Systems – IDS إجراءات مراقبة، وتنشئ تنبيهات عندما تكتشف موقفاً مشبوهاً، وبعبارة أخرى، تبحث أنظمة اكتشاف الاختراق عن أعراض لأحداث أمنية محتملة.[7]

لدينا أنظمة معتمدة على المعلومات المحلية، فمثلاً، تجمع معلومات من سجلات النظام، وتراقب التغييرات في نظام الملفات، أو في إعدادات الخدمات الشائعة. وهناك أنظمة أخرى معتمدة على الشبكة وتحقق من أنه لا يوجد سلوك غريب، كالخداع بالتظاهر كعنوان معروف، والتحكم بسير البيانات المشبوهة، والهجمات المحتملة لمنع خدمة، واكتشاف سير البيانات بشكل مكثف نحو خدمة معينة، وضبط أنه لا توجد واجهات شبكة في وضع مشبوه (من أعراض المتنصتات أو ملتقطات الحزم).

مثال

من الأمثلة على أدوات أنظمة اكتشاف الاختراق: Logcheck (للتحقق من السجلات)، و TripWire (حالة النظام عبر تنفيذ حسابات md5 على الملفات)، و AIDE (نسخة مجانية من TripWire)، و Snort (أداة اكتشاف اختراق للتحقق من حالة شبكة بأكملها).

---

7 تسمح لنا أنظمة اكتشاف الاختراق باكتشاف الدخلاء في نفس اللحظة التي يقومون فيها باستخدام مواردنا أو استغلال أنظمتنا بحثاً عن مشاكل أمنية.

7 الحماية بالترشيح عبر المغلِّفات (wrappers) وجدران الحماية

إن مغلِّفات TCP (أو TCP wrappers) هي برامج تعمل كوسيط بين طلبات المستخدمين لخدمة ما ومراقبات الخوادم التي تقدم الخدمة. تأتي معظم التوزيعات مع الـ wrappers مفعلة ونحن نضبط مستويات الوصول. وتستخدم المغلفات مع inetd و xinetd من أجل حماية الخدمات التي توفرها.

وبشكل أساسي، تستبدل المغلفاتُ مراقباتِ الخدمات إلى أخرى تعمل كوسيطة (تدعى tcpd، وموجودة في العادة في /ust/sbin/tcpd/). عندما يستقبل هذا البرنامج الطلب، يتحقق من المستخدم ومصدر الطلب، من أجل التحقق مما إذا كان يسمح مغلِّف الخدمة باستخدامها أم لا. وتتضمن أيضاً إمكانية إنشاء تقارير، أو للإعلام عبر البريد عن المحاولات المحتملة للوصول، ومن ثم تشغل المراقب المناسب المرتبط بالخدمة.

على سبيل المثال، لنفترض وجود المدخلة التالية في inetd:

```
finger stream tcp nowait nobody /usr/etc/in.fingerd in.fingerd
```

نغيرها إلى:

```
finger stream tcp nowait nobody /usr/etc/in.fingerd
```

وبالتالي عند وصول طلب، يتم التعامل معه عبر المراقب tcpd الذي سيكون مسؤولاً عن التحقق من الوصول (للمزيد من المعلومات المفصلة، انظر إلى صفحات man tcpd).

هناك أيضاً طريقة بديلة لـ TCP wrapper، وتتكون من تصريف التطبيق الأصلي مع مكتبة المغلِّف wrapper. وبهذه الطريقة لا يجب على التطبيق أن يكون موجوداً في inetd، ويمكننا أن نتحكم به كما في الحالة الأولى بالإعداد الذي سنناقشه فيما يلي.

يتم التحكم بنظام التغليف عبر الملف etc/hosts.deny/، والذي نحدد فيه أي خدمات نمنع وعن مَن، باستخدام خيارات مثل صدفة صغيرة لحفظ المعلومات في المحاولة، والملف etc/hosts.allow/ الذي نضع فيه الخدمة التي ننوي استخدامها، متبوعة بقائمة المسموح لهم باستخدام الخدمة (ولاحقاً في الدرس التعليميّ، سنلقي نظرة على مثال صغير). لدينا أيضاً

الأمر tcpdchk الذي يختبر إعداد ملفات المضيفين (انظر إلى دليل hosts_access و hosts_options) للتحقق من أنها صحيحة، وبعبارة أخرى، تتفحص الإعدادات. والأمر الآخر المفيد هو tcpdmatch الذي نعطيه اسم خدمة وعميلاً محتملاً (المستخدم و/أو المضيف)، وتخبرنا بما سيقوم به النظام في هذه الحالة.

7.1 جدران الحماية

جدار الحماية نظام أو مجموعة من الأنظمة التي تنفذ سياسات تحكم بالوصول بين الشبكات. يمكن تنفيذ جدار النار في تطبيق يعمل على جهاز مستقل أو على جهاز خاص مصمم لحماية حاسوب أو أكثر.

وبشكل عام، يكون لدينا إما تطبيق جدار حماية لحماية جهاز معين متصل مباشرة بالإنترنت (مباشرة أو عبر مزود خدمة)، أو يمكن استخدام جهاز أو عدة أجهزة مصممة لهذا الغرض على شبكتنا من أجل حماية شبكتنا الداخلية.

والحل الأفضل تقنياً هو أن يكون لدينا حاسوب ببطاقتي شبكة أو أكثر يعمل على عزل الشبكات المختلفة المتصلة (أو أجزاء الشبكة)، وبتلك الطريقة يكون جدار الحماية على الجهاز (أو إذا كان عتاداً خاصاً) مسؤولاً عن توصيل حزم الشبكة وتحديد أيها يمكن أن تمر وإلى أي شبكة.

عادة ما يكون هذا النوع من جدران الحماية مضمناً داخل الموجه لربط حزم الشبكات المختلفة. وهناك إعداد آخر شائع، وهو جدار الحماية من الإنترنت، فعلى سبيل المثال، توجد بطاقتا شبكة: نرسل ونستقبل البيانات مع الإنترنت عبر إحداها، وبالتالي نتخلص من سير البيانات غير الموجهة إلينا ونتحكم بسير البيانات الصادرة إلى الإنترنت، في حال لم نكن نرغب بالوصول إلى موافيق معينة، أو إذا كنا نشكّ بأن هناك تسريبات محتملة للمعلومات بسبب هجمة ما. وهناك احتمال ثالث ويكون في جهاز منفرد متصل بالإنترنت عبر بطاقة شبكة واحدة، سواء مباشرة أو عبر مزود خدمة. وفي هذه الحالة، يكون اهتمامنا فقط بحماية جهازنا من الدخلاء ومن حركة البيانات غير المرغوب بها ومن البيانات التي يشتبه بكونها تمر ضمن عملية سرقة للبيانات.

وبعبارة أخرى، في كل هذه الحالات يمكننا أن نرى أنه يمكن أن يكون لجدار الحماية إعدادات واستخدامات مختلفة تعتمد على ما إذا كان برمجياً أم لا، وإذا كان للجهاز بطاقة شبكة واحدة أو أكثر، وإذا كانت تحمي جهازاً منفرداً أو مجموعة من الأجهزة.

وبشكل عام، يسمح جدار الحماية للمستخدم أن يحدد مجموعة من السياسات (أي من الأجهزة يمكن أن يتم وصلها لعمل ذلك، أو أي الأجهزة يمكن أن تستقبل معلومات، وأي نوع من المعلومات)، وذلك عبر التحكم بمنافذ TCP/UDP الصادرة والواردة. تأتي بعض جدران الحماية مع سياسات مضبوطة مسبقاً، بينما تسمح غيرها بأن يتم تفصيل كل الخيارات (الأجهزة، الموافيق، المنافذ، إلخ).

وهناك تقنية أخرى ذات علاقة، وهي ترجمة عناوين الشبكة Network Address Translation، ويشار إليها اختصاراً بكلمة "نات" NAT. تقدم هذه التقنية مسلكاً إلى إخفاء عناوين الشبكة المستخدمة في الشبكة الخاصة عن الإنترنت، مع الإبقاء على الوصول من الأجهزة. إن أحد الأساليب المعتادة ما يعرف بالاسم masquerading. باستخدام NAT masquerading، يمكن لجهاز أو أكثر على الشبكة أن يظهروا كعنوان IP واحد للناظر من الخارج. يسمح هذا للعديد من الأجهزة أن تتصل بجهاز وحيد للاتصالات الخارجية؛ فعلى سبيل المثال، في حالة موجهات ADSL في المنزل التي تسمح لمجموعة من الأجهزة أن تتصل دون الحاجة لأن يعطي مزود الخدمة عدة عناوين IP. تقدم موجهات ADSL في العادة نوعاً من NAT masquerading، وإمكانيات جدار الحماية أيضاً. ومن الشائع أيضاً استخدام تركيبة من كلتي التقنيتين. في هذه الحالة، يكون إعداد الشبكة الداخلية الخاصة التي نرغب بحمايتها متوفراً، وهو في ذلك مثل جهاز جدار الحماية (في الحالات التي رأيناها أعلاه).

4.3 Netfilter: IPtables

تقدم نواة لينكس (منذ الإصدار 2.4) نظاماً فرعياً للترشيح يدعى Netfilter، والذي يقدم مزايا ترشيح الحزم إضافة إلى NAT. يسمح هذا النظام باستخدام واجهات الترشيح المختلفة، والأكثر شيوعاً بينها هي Iptables. قبل ذلك (في الإصدار 2.2)، كانت توفر مرشحاً آخر يدعى ipchains، وكان للنظام سياق مختلف (ولكنه شبيه). أما النواة 2.0 (الأقدم منها)، فقد كانت تستخدم نظاماً مختلفاً يدعى ipfwadm. سنتعامل هنا (وفي الأمثلة اللاحقة) مع Netfilter/IPtables فقط (وبعبارة أخرى، مع ما هو مستخدم الآن).

تسمح واجهة الأمر iptables بالقيام بالمهام المختلفة لضبط القواعد التي تؤثر على نظام الترشيح: كإنشاء التقارير، و

وظائف توجيه الحزم القبلية والبعدية، و NAT، وتمرير المنافذ.

يتم تشغيل الخدمة بالامر: etc/init.d/iptables start/، إذا لم تكن مضبوطة مسبقاً في مستوى التشغيل.

يعرض الأمر iptables -L قائمة بالقواعد الفعالة في تلك اللحظة في كل من سلاسل القواعد. وإذا لم تكن مضبوطة مسبقاً، فستقبل مبدئياً كل الحزم في سلاسل الدخل والخرج والتمرير.

يتعامل نظام Iptables مع الجداول كطبقة عليا. يحوي كل منها سياسة مختلفة، تحوي كل منها قواعد مختلفة أيضاً. الجداول الثلاثة التي لدينا هي: Filter و NAT و Mangled. الأول لترشيح القواعد نفسها، والثاني لترجمة العناوين داخل النظام الذي يستخدم NAT، و الثالث – الأقل استخداماً – يخدم في تحديد بعض خيارات التحكم بالحزم وكيفية إدارتها. وعلى وجه التحديد، إذا كان لدينا نظام متصل مباشرة بالإنترنت، فبشكل عام لن نستخدم سوى جدول الترشيح Filter. إذا كان الجهاز ضمن شبكة خاصة عليها أن تمرر عبر موجه أو عبّارة أو خادم وسيط (أو تجميعة منها)، فعلى الأرجح سيكون لدينا نظام NAT أو IP masquerading، إذا كنّا نضبط الجهاز للسماح بالوصول الخارجي، فسيكون علينا تعديل جدول NAT وجدول الترشيح (Filter). إذا كان الجهاز ضمن نظام شبكة خاصة، ولكنه من الأجهزة الداخلية، فسيكون تحرير جدول الترشيح كافياً، ما لم يكن خادماً يترجم عناوين الشبكة لجزء آخر من الشبكة.

عندما تصل حزمة إلى الشبكة، فسينظر جدار الحماية فيما إذا كانت القواعد في جدول NAT، للتأكد مما إذا كانت هناك عناوين ضمن الشبكة الداخلية تحتاج إلى ترجمة (عادةً لا تكون العناوين مرئية خارجياً)؛ ثم ستنظر في قواعد جدول الترشيح من أجل تحديد ما إذا كان سيتم السماح للحزم بأن تمرر، أو إذا لم تكن لنا وكان لدينا قواعد للتمرير لمعرفة إلى أين يتم تمريرها. ومن الناحية الأخرى، عندما تُنشئ عملياتنا حزماً، فستتحكم قواعد الإخراج لجدول الترشيح بما إذا كانت ستسمح لها أم لا، وإذا كان هناك نظام NAT فستترجم القواعدُ العناوين من أجل عمل masquerade لها. هناك في العادة سلسلتان في جدول NAT: ما قبل التوجيه، وما بعد التوجيه. في الأولى يكون على القاعدة أن تحدد ما إذا كان يجب توجيه الحزمة، وإذا كان كذلك، فإلى أي وجهة سيتم توجيهها. وفي الثانية، يتم تحديد إذا كان يسمح للحزمة بالدخول أم لا (إلى الشبكة الخاصة مثلاً) بشكل نهائي. وهناك أيضاً سلسلة إخراج للبيانات المُنشأة محلياً الخارجة إلى الشبكة الخاصة، حيث أن ما قبل التوجيه لا

يتحكم بهذا (لمزيد من المعلومات، راجع man iptables).

سنعلق لاحقاً على بعض النواحي والأمثلة على ضبط جدول الترشيح (للجداول الأخرى، يمكننا أن نتفقد المراجع ذات العلاقة).

من المعتاد ضبط جدول الترشيح كمجموعة من القواعد التي تحدد ما يتم فعله داخل سلسلة معينة، كالثلاثة السابقة (الدخل والخرج والتمرير). وفي العادة، سنقوم بتحديد:

```
iptables -A chain -j target
```

حيث تكون قيمة chain إما input أو output أو forward، و target الوجهة التي سيتم تمرير الحزمة إليها والتي تتوافق مع القاعدة. يضيف الخيار A- القاعدة إلى ما هو موجود. علينا أن نكون حذرين هنا، لأن الترتيب مهم. علينا أن نضع القواعد الأقل صرامة في البداية، علماً بأننا إذا وضعنا قاعدة تتخلص من الحزم في البداية، فحتى لو كانت هناك قاعدة أخرى، فلن يتم أخذها بعين الاعتبار. يمكن أن يُستخدم الخيار j- لتحديد ما سنفعله مع الحزم، وعلى وجه التحديد، القبول accept أو الرفض reject أو التجاهل drop. من المهم التمييز بين reject و drop. ففي الأولى نرفض الحزمة ونخبر في الوضع الاعتيادي المرسل بأننا رفضنا محاولة الاتصال (وعادة ما تكون لحزم من نوع ICMP). أما في الثانية – التجاهل drop – فببساطة "نضيع" الحزمة كما لو لم تكن موجودة، ولن نرسل أي نوع من أنواع الردّ. وهناك target آخر مستخدم، وهو log، وذلك لإرسال الحزمة إلى نظام التقارير. وعادة – في هذه الحالة – هناك قاعدتان، إحداها مع log، والأخرى اعتيادية بـ accept أو reject أو drop، بحيث يصير من الممكن إرسال المعلومات عن الحزم المقبولة أو المرفوضة أو التي تم تجاهلها إلى التقارير.

عند إدخال القاعدة، فنحن نستخدم أيضاً الخيار I- (أي insert)، الذي يشير إلى الموضع، على سبيل المثال:

```
iptables -I INPUT 3 -s 10.0.0.0/8 -j ACCEPT
```

والتي تخبرنا بأنه يفترض أن يتم وضع القاعدة في المكان الثالث من سلسلة INPUT (الدخل)؛ وأن الحزم (j-) التي تأتي من (المصدر source، ويحدد بالخيار s-)، من الشبكة الفرعية 10.0.0.0، بالقناع 255.0.0.0، سيتم قبولها. وبالخيار D – وبشكل مشابه لما سبق – يمكننا أن نحذف رقم القاعدة أو القاعدة نفسها، كما هو محدد أدناه، بحذف القاعدة الأولى من السلسلة، أو القاعدة التي ذكرناها:

```
iptables -D INPUT 1
iptables -D INPUT -s 10.0.0.0/8 -j ACCEPT
```

هناك أيضاً قواعد يمكن استخدامها لتحديد سياسة مبدئية للحزم (بالخيار P)؛ سيتم عمل نفس الشيء بكل الحزم. على سبيل المثال، سنقرر عادة تجاهل كل الحزم مبدئياً ومن ثم تفعيل ما نحتاجه منها؛ وبشكل مشابه، سنتجنب عادة تمرير الحزم إذا كانت إذا لم تكن ضرورية (إذا لم نكن نتصرف من الموجّه)، فيمكن الإعلان عن ذلك كما يلي:

```
iptables -P INPUT DENY
iptables -P OUTPUT REJECT
iptables -P FORWARD REJECT
```

يفعل هذا السياسات المبدئية التي تتكون من تجاهل أي حزم قادمة وعدم السماح بإرسال أو إعادة إرسال الحزم. سيكون بإمكاننا الآن إضافة القوعد التي تؤثر على الحزم التي نرغب باستخدامها، بتحديد أي موافيق ومنافذ ومصادر أو وجهات نرغب بالسماح بها أو تجنبها. سيكون هذا صعباً حيث سنحتاج لمعرفة كل المنافذ والموافيق التي تستخدمها برمجياتنا أو خدماتنا. وهناك استراتيجية أخرى، وهي ترك الخدمات الضرورية فقط مفعلة وتفعيل الوصول إلى الخدمات للأجهزة المرغوبة عبر جدار الحماية.

بعض هذه الأمثلة في جدول التوجيه يمكن أن تكون:

```
1) iptables -A INPUT -s 10.0.0.0/8 -d 192.168.1.2 -j DROP
2) iptables -A INPUT -p tcp --dport 113 -j REJECT --reject-with-tcp-reset
3) iptables -I INPUT -p tcp --dport 113 -s 10.0.0.0/8 -j ACCEPT
```

حيث:

1) نتجاهل الحزم التي تأتي من 10. x.x.x والمرسلة إلى 192.168.1.2.

2) نرفض حزم TCP المرسلة إلى المنفذ 113، بإرجاع ردّ من نوع tcp-reset.

3) نفس الحزم التي في النقطة 2، لكن الحزم التي تأتي من 10. x.x.x يتم قبولها.

وفيما يتعلق بأسماء الموافيق والمنافذ، يستخدم نظام iptables المعلومات التي تقدمها الملفات /etc/services و /etc/protocols، ويمكننا تحديد المعلومات (الميفاق أو المنفذ) سواء عبر الأرقام أو بالأسماء (نتأكد في هذه الحالة أن

معلومات الملفات صحيحة وأنه لم يتم تعديلها، من طرف مهاجم مثلاً).

يتم في العادة بناء إعدادات iptables عبر استدعاءات متتالية لأمر iptables مع القواعد. يُنشئ هذا حالة من القواعد الفعالة التي يمكن مراجعتها بالأمر iptables، إذا كنا نرغب بحفظها بحيث تبقى بشكل دائم، فيمكننا عمل ذلك في فيدورا بالأمر:

```
/etc/init.d/iptables save
```

ويتم حفظها في:

```
/etc/sysconfig/iptables
```

يمكننا في دبيان تنفيذ:

```
/etc/init.d/iptables save name-rules
```

علينا أن نكون حذرين وأن نتأكد من أن المجلد var/log/iptables/ موجودة مسبقاً، حيث أنها المكان الذي تحفظ فيه الملفات، سيكون name-rules ملفاً في المجلد.

يمكننا بالأمر (etc/init.d/iptables load/) أن نحمل القواعد (علينا في دبيان أن نحدد اسم ملف القواعد)، رغم أن دبيان تدعم بعض الأسماء المبدئية للملفات – والتي تكون مفعلة للقواعد الاعتيادية (التي يتم استخدامها عند بدء الخدمة) – وغير مفعلة للتي ستبقى في عند تعطيل (أو توقف) الخدمة. وهناك طريقة أخرى شائعة الاستخدام أيضاً، وهي وضع القواعد في ملف نصّ برمجي مع استدعاءات iptables الضرورية واستدعائها – على سبيل المثال – بوضعها في مستوى التشغيل الضروري، أو برابط إلى النص البرمجي في /etc/init.d/.

4.4 حزم جدران الحماية في التوزيعات

فيما يتعلق بأدوات الإعداد الآلية نوعاً ما لجدار الحماية، هناك العديد من الإمكانات، لكن علينا أن نتذكر أنها لا توفر عادة نفس مزايا الإعداد اليدوي لـ iptables (وهي عملية مستحسنة في معظم الحالات). بعض الأدوات هي:

- lokkit: يمكن للمستخدم في فيدورا/ردهات – في مستوى أساسيّ جدًّا – أن يختار مستوى الأمن المرغوب (عال، أو

متوسط، أو منخفض). وبعد ذلك يتم عرض الخدمات التي ستتأثر، ويمكننا أن نتركها، أو أن نستمر إلى الخدمة لتغيير الإعداد المبدئي. الآليات المستخدمة أدناه هي في iptables. يمكن رؤية الإعداد النهائي للقواعد التي تم عملها في etc/sysconfig/iptables الذي تقرأه خدمة iptables التي يتم تحميلها عند الإقلاع، أو عند الإيقاف والبدء باستخدام etc/init.d/iptables/، بالخيار start أو stop. من الممكن أيضاً تثبيتها في دبيان، لكن يجب ترك إعداد القواعد في etc/default/lokkit-1، والنص البرمجي في etc/init.d/lokkit-1/. هناك أيضاً إصدار رسومي يدعى gnome-lokkit.

- Bastille: هذا برنامج أمني تعليمي كامل نوعاً ما، يشرح الإعدادات الأمنية المستحسنة المختلفة، وكيفية تنفيذها خطوة بخطوة، وتشرح أيضاً إعداد جدار الحماية (البرنامج تفاعلي). تعمل في العديد من التوزيعات، بما فيها فيدورا ودبيان.
- Fwbuilder: أداة يمكن استخدامها لإنشاء قواعد جدار الحماية باستخدام واجهة رسومية. يمكن استخدامها في العديد من أنظمة التشغيل (جنو/لينكس، بما في ذلك فيدورا ودبيان، و OpenBSD، و MacOS)، بأنواع مختلفة من جدران الحماية (بما فيها iptables).
- Firestarter: أداة رسومية (جنوم) لإنشاء جدار حماية. وهو كامل جداً، ويدير عملياً كل إمكانيات iptables، لكن كذلك له مساعدين يجعلون من السهل ضبط جدار الحماية من البديهيات.

وفي العادة، تستخدم كل من هذه الحزم نظام قواعد يحفظ في ملف إعداداتها، وتعمل عادة كخدمة أو كنص برمجي يتم تنفيذه في ستوى الإقلاع المبدئي.

4.5 وفي النهاية، أشياء يجب أخذها بعين الاعتبار

حتى وإن ضبطنا جدران حماية بشكل جيد، فعلينا أن نتذكر أنها ليست حماية أمنية مطلقة، حيث أن هناك هجمات معقدة يمكنها تخطي جدران الحماية وتشويه البيانات لخلط الأمر. وإضافة إلى ذلك، تحتاج الاتصالات الحديثة أحياناً لأن تجبرنا على إنشاء برمجيات تتخطى جدران الحماية[8]:

8 للمزيد، ابحث في الوب عن hole punching.

- تجعل تقنيات مثل IPP أو ميفاق الطباعة الذي يستخدمه CUPS، أو WebDAV، وموافيق تأليف وعمل إصدارات للمواقع تخطيَ إعدادات جدار الحماية ممكناً (أو ضرورياً).

- تستخدم كثيراً تقنية تُدعى tunneling (كالموافيق التي ذكرت أعلاه وغيرها). ببساطة، تُضَمِّن هذه التقنية الموافيق غير المسموح لها على أنها أخرى مسموح بها؛ على سبيل المثال، إذا كانت جدران الحماية تسمح فقط لسير بيانات HTTP بالمرور عبرها (المنفذ 80 مبدئيًّا)، فمن الممكن كتابة عميل وخادم (كل منها في جهة من جدار الحماية) قادرين على التخاطب عبر ميفاق معروف لكليهما، ولكن يتم تحويل حزم الشبكة إلى حزم HTTP معيارية، مما يعني أنه يمكن للبيانات تخطي جدار الحماية.

- الأكواد المحمولة على الوِب (ActiveX و Java و JavaScript) تتخطى جدران الحماية، وبالتالي يصعب حماية الأنظمة إذا كانت ضعيفة ضد الهجمات على أي ثغرات مفتوحة يتم اكتشافها.

وبالتالي، رغم كون جدران الحماية حلول جيدة جداً لمعظم النواحي المتعلقة بالأمن، فيمكن دائماً أن تحوي ثغرات وتدع البيانات التي تعتبرها صالحة تمر، والتي تتضمن بالتالي مصادر أخرى محتملة للهجمات أو الثغرات. وفيما يتعلق بالأمن، علينا أن لا نستخدم أبداً (أو نعتمد على) حلّ وحيد ونظن أنه يحمينا من أي شيء؛ من الضروري فحص المشكلات المختلفة، وتقديم حلول تكتشف أيّ مشكلات على الفور، وإنشاء سياسات منع تحمي النظام قبل وقوع أيّ أضرار.

8 الأدوات الأمنية

يمكن اعتبار بعض هذه الأدوات كأدوات لاختراق الأجهزة الأخرى أيضاً. لهذا، يُنصح باختبارها على أجهزة في شبكتنا الخاصة، علينا أن لا نقوم بذلك على عناوين الغير أبداً، حيث يمكن أن يفسروا هذه الاختبارات على أنها محاولات اختراق، ويتم تحميلنا أو تحميل مزود خدمة الإنترنت لدينا المسؤولية عنها، ويمكن أن يتم استدعاء الجهات المختصة للتحقيق معنا وقطع اتصالنا.

سنناقش الآن بشكل موجز بعض الأدوات والطرق التي يمكن استخدامها فيها:

أ. TripWire: وهي أداة تحتفظ بقاعدة بيانات بمجاميع للتحقق من الملفات الهامة في النظام.

يمكن أن تفيد كنظام اكتشاف للاختراق IDS. يمكننا استخدامها لأخذ snapshot عن النظام، بحيث نتمكن من فحص أي تغييرات تم القيام بها والتأكد من أنه لم يتم تخريبه من طرف مهاجم. الهدف هنا هو حماية الملفات في الجهاز نفسه، وتجنب أي تغييرات تحدث، كتلك التي يمكن أن تكون قد سببها rootkit مثلاً. ولهذا، عندما نشغل الأداة مرة أخرى، يمكننا فحص كل التغييرات ومقارنتها بآخر مرة تم الفحص فيها. علينا أن نختار مجموعة من الملفات الهامة في النظام، أو التي تعد مصدراً محتملاً للهجمات. TripWire مملوك، لكن هناك أداة مكافئة حرة ومفتوحة المصدر تُدعى AIDE.

ب. nmap،وهي أداة تفحص المنافذ في الشبكات الكبيرة. يمكنها البحث من جهاز واحد إلى أجزاء من الشبكة، وتقدم العديد من أوضاع البحث، معتمدة على حماية النظام. وتقدم أيضاً تقنيات يمكنها تحديد نظام التشغيل المستخدَم على الأجهزة البعيدة. يمكن استخدام حزم TCP و UDP مختلفة لفحص الاتصالات. هناك واجهة رسومية تعرف بالاسم xnmap.

```
Nmap Front End v3.00
File  Output  View  BETA Options  Help
Host(s): 127.0.0.1   Scan.  Exit
Scan Options:   General Options:
connect()  Don't Resolve  TCP Ping  Fragmentation
SYN Stealth  Fast Scan  TCP&ICMP  Get Identd Info
Ping Sweep  Range of Ports  ICMP Ping  Resolve All
UDP Port Scan  Don't Ping  OS Detection
FIN Stealth
Bounce Scan:  Use Decoy(s):  Input File:  Send on Device
Output from: nmap -sT -O 127.0.0.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
 Interesting ports on localhost.localdomain (127.0.0.1):
(The 1596 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
111/tcp    open        sunrpc
631/tcp    open        ipp
6000/tcp   open        X11
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.204 days (since Tue Sep 23 10:45:47 2003)

Nmap run completed -- 1 IP address (1 host up) scanned in 7
seconds
```

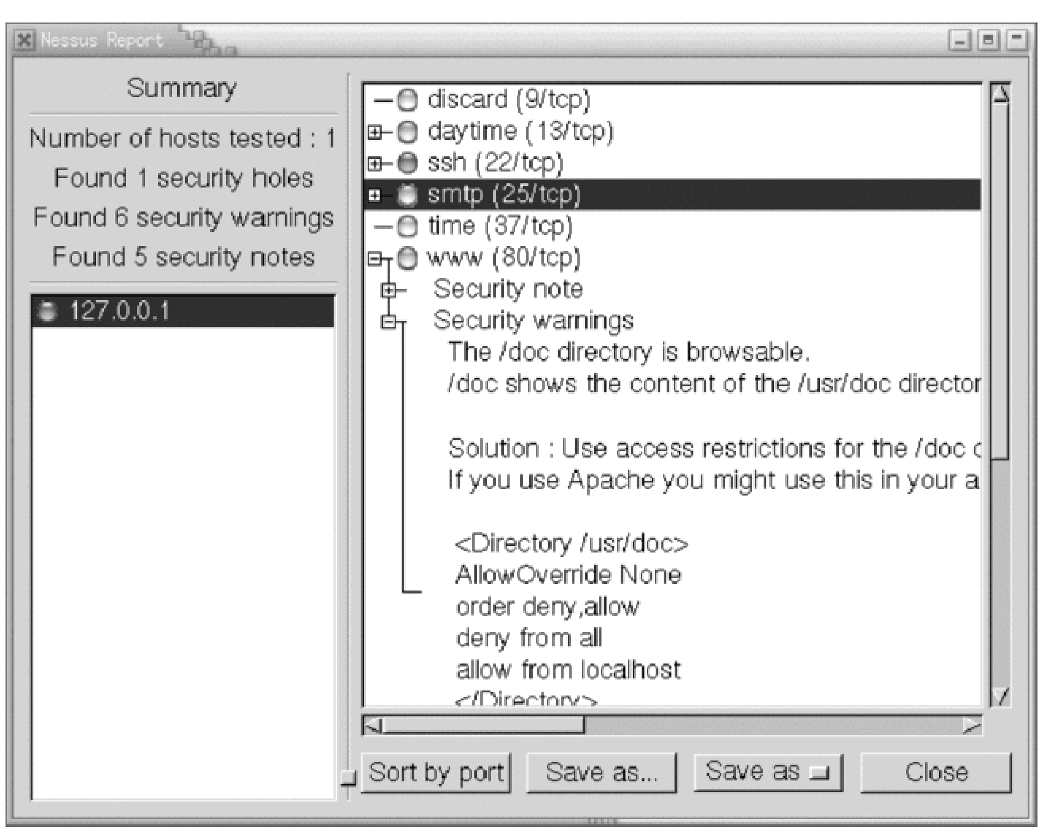
شكل 2: xnmap يحلل الخدمات المحلية

جـ. WireShark (وكان يُعرف قديماً بالاسم Ethereal): وهو محلل موافيق يلتقط البيانات المارة في الشبكة (يعمل كمتنصّت sniffer). يمكن استخدامه لالتقاط البيانات، ورؤية إحصائيات وبيانات الحزم المنفردة ومجموعات الحزم، سواء عبر المصدر، أو الوجهة، أو المنفذ، أو الميفاق. ويمكنه حتى إعادة إنشاء حركة البيانات من جلسة كاملة من ميفاق التحكم بالنقل TCP.

د. **Snort:** وهو نظام لاكتشاف الاختراق يجعل من الممكن تحليل سير البيانات في الوقت الحقيقي وحفظ تقارير الرسائل. يمكن أن يُستخدَم لتحليل الموافيق والبحث عن أنماط (الميفاق، المصدر، الوجهة، إلخ). يمكن أن يُستخدم لاكتشاف أنواع عدّة من الهجمات. وبشكل أساسي، يُحلل سير البيانات التي قد تشابه هجمة. يستخدم النظام مجموعة من القواعد إما لإنشاء

سجل بالأوضاع (تقرير – log)، أوتنبيه المستخدم (تحذير alert)، أو رفض المعلومات (تجاهل drop).

هـ. Nessus: يكتشف أية ثغرات معروفة (عبر اختبار تقنيات الاختراق المختلفة) وتدعّم أفضل الخيارات الأمنية لما هو مكتشف. هذا برنامج مجزّأ يتضمن مجموعة من الإضافات (أكثر من أحد عشر ألفاً) للقيام بالتحاليل المختلفة. تستخدم معمارية العميل – الخادم، بعميل رسومي لعرض النتائج، والخادم الذي يقوم بالاختبارات المختلفة على الأجهزة. وله القدة على فحص شبكات بأكملها. يُنشئ Nessus تقارير عن النتائج، التي يمكن تصديرها بصيغ مختلفة (HTML مثلاً). وحتى 2005، كانت Nessus 2 أداة حرّة، لكن الشركة قررت جعلها مملوكة في الإصدار الثالث. ما زال Nessus 2 مستخدماً في جنو/لينكس، حيث احتفظ بترخيص GPL ومجموعة من الإضافات التي تحدّث دورياً. Nessus 2 – الأداة المملوكة لجنو/لينكس – أقوى وأوسع استخداماً، حيث أنها من أكثر الأدوات الأمنية شيوعاً وهناك في العادة إصدار مجاني متاح مع إضافات بتحديثات أقل من تلك التي في الإصدار غير المجاني.



شكل 3: عميل Nessus يعرض تقرير الثغرات والحلول الممكنة

يمكننا أن نجد كثيراً من الأدوات الأخرى المتاحة. ومن الأماكن المناسبة للبدء http://sectools.org، حيث يقدم مصممو Nmap قائمة بالأدوات الشائعة، كما يصوت عليها المستخدمون (وهي الآن قائمة قديمة، لكنها تحوي أدوات مفيدة).

9 تحليل التقارير

بمراقبة ملفات التقارير (أو السجلات – logs)، يمكننا أن نكون بسرعة فكرة عن الحالة العامة للنظام، إضافة إلى الأحداث الأخيرة، اكتشاف أيّة اختراقات (أو محاولات اختراق) غير طبيعية. لكن يجب أن نتذكر أنه إذا كان هناك اختراق بالفعل، فيمكن أن تكون التقارير قد حُذفت أو عُدِّلت. ستكون معظم ملفات التقارير في المجلد /var/log/.

كثير من الخدمات يمكن أن يكون لها ملفات تقارير خاصة بها، والتي يتم إنشاؤها بالعادة أثناء الإعداد (عبر ملف الإعدادات ذي العلاقة). يستخدم معظمها إمكانيات التقارير المضمنة في syslog عبر المراقب syslogd. سيكون الإعداد في /etc/syslog.conf. يتم إنشاء هذا التقرير عادة اعتماداً على مستويات الرسائل: هناك أنواع مختلفة للرسائل تعتمد على أهميتها. وفي العادة، تظهر مستويات مثل debug, info, err, notice, crit, alert, emerge، التي سيكون ترتيب الرسائل فيها حسب الأهمية كما يلي. وعادة يتم إرسال الرسائل إلى التقرير /var/log/messages، لكن يمكن أن يتم ضبط النظام بحيث يذهب كل نوع منها إلى ملف مختلف، ومن الممكن أيضاً تحديد من أنشأها، وتكون عادة أنظمة النواة والبريد والأخبار والاستيثاق، إلخ.

ونتيجة لذلك، من المناسب تفحص (وعلى أيّة حال، تطويع) إعداد syslog بحيث يتم تحديد التقارير التي يمكن أن نجد/ننشئ فيها المعلومات. وهناك نقطة أخرى هامة، وهي التحكم بنموها، اعتماداً على أيها فعالة، والعمليات (والخدمات) التي يتم تنفيذها في النظام. يمكن أن تنمو التقارير بسرعة كبيرة. في دبيان وفيدورا، يمكن التحكم بهذا عبر logrotated، وهو مراقب يقوم بعمل نسخٍ دوريّ ويضغط التقارير الأقدم؛ من الممكن إيجاد الإعداد العام في /etc/logrotate.conf، رغم أن بعض التطبيقات تضع إعدادات معينة يمكن إيجادها في المجلد /etc/logrotate.d/.

في النقاط التالية، سنناقش بعض ملفات التقارير التي يفترض أن نأخذها بعين الاعتبار (وربما هي الأكثر استخداماً):

1) /var/log/messages: هو ملف التقارير المبدئي لمراقب syslogd، ولكن سيكون علينا تفحص إعداداته، في حال تم نقله إلى مكان آخر، أو سيكون هناك ملفات عدة منها. يحوي هذا الملف طيفاً واسعاً من الرسائل من مصادر عديدة (خدمات ومراقبات عديدة، أو النواة نفسها)؛ يجب التحقق من أي شيء لا يبدو اعتيادياً. إذا كان هناك اختراق، فيفترض أن يتم التحقق من تاريخ الاختراق والملفات ذات العلاقة.

2) var/log/utmp/: يحوي هذا الملف معلومات ثنائية لكل مستخدم يعمل في تلك اللحظة. وهو مفيد لمعرفة الوالجين في النظام. يستخدم الأمر who هذا الملف لتقديم المعلومات.

3) var/log/wtmp/: في كل مرة يلج أو يخرج المستخدم فيها من النظام، أو يعاد فيها تشغيل الجهاز، يتم حفظ مدخلة في هذا الملف. هذا ملف ثنائي يحصل منه الأمر الأخير على المعلومات؛ يسجل الملف المستخدمين الذين يلجون أو يخرجون من النظام، والوقت والمكان الذي تم منه الاتصال. قد يكون من المفيد معرفة المكان (والحساب) الذي تم منه الاختراق، واكتشاف استخدام حسابات مشبوهة. هناك أيضاً تنوع في الأوامر يُدعى lastb، الذي يعرض محاولات الولوج التي لم يتم التحقق منها بشكل صحيح، ويتم استخدام var/log/btmp/ (قد تضطر لإنشائه إذا لم يكن موجوداً) يمكن إرسال نفس أخطاء الاستيثاق إلى السجل auth.log. وبطريقة مشابهة، يستخدم الأمر lastlog ملفاً آخر – وهو var/loglastlog/ – للتحقق من الاتصال الأخير لكل من المستخدمين.

4) var/log/secure/: وهو مستخدم عادة في فيدورا لإرسال رسائل tcp wrapper (أو جدران الحماية). وفي كل مرة يتم فيها إنشاء اتصال إلى خدمة inetd أو xinetd، يتم إضافة رسالة تقرير إلى هذا الملف. يمكننا البحث عن محاولات الاختراق في الخدمات التي لا تُستخدَم عادة أو في الأجهزة غير المعروفة التي تحاول الاتصال.

في نظام السجلات، هناك أمر آخر يفترض أن يتم تفقده، وهي أن التقارير في المجلد var/log/ يمكن أن يكتب فيها الجذر فقط (أو المراقبات المرتبطة بالخدمات). وإلا فسيكون بإمكان المهاجم تغيير المعلومات في السجلات. إضافة إلى ذلك، إذا تمكن مهاجمون من الوصول إلى الجذر، فيمكنهم حذف كل آثارهم.

سنقوم الآن بتحليل العمليات المذكورة أعلاه في نظام دبيان، وذلك لتحسين الإعدادات الأمنية. سنتفقد في البداية أن جهازنا به اتصال بالشبكة. لعمل هذا، سنستخدم أداة nmap كماسح منافذ، وذلك بالأمر التالي (من الجذر):

```
nmap -sTU -O localhost
```

نحصل على ما يلي:

```
root@machine:˜# nmap -sUT -O localhost
starting nmap 3.27 (www.insecure.org/nmap/) at 2003-09-17 11:31 CEST
Interesting ports on localhost (127.0.0.1):
```

(المنافذ الـ 3079 التي تم مسحها ولم تُعرَض في ما يلي هي منافذ مغلقة)

```
Port      State     Service
9/tcp     open      discard
9/udp     open      discard
13/tcp    Open      daytime
22/tcp    Open      smtp
25/tcp    open      time
37/tcp    open      time
37/udp    open      http
80/tcp    open      sunrpc
111/tcp   open      sunrpc
111/udp   open      auth
113/tcp   open      ipp
631/tcp   open      unknown
728/udp   open
731/udp   open      netviewdm3
734/tcp   open      unknown
Remote operating system guess: Linux kernel 2.4.0-2.5.20
Uptime 2.011 days (since Mon Sep 15 11:14:57 2003)
Nmap run completed – 1 IP address (1 host up) scanned in 9.404 seconds
```

يمكننا أن نرى أن عدداً كبيراً من الخدمات التي تم اكتشافها (ويمكن أن يكون هناك أكثر من هذا اعتماداً على الجهاز: telnet, ftp, finger، ...)، في كل من ميفاقي TCP و UDP. بعض الخدمات، مثل discard و daytime و

time، يمكن أن تكون مفيدة في حالات معينة، لكن يفترض أن لا تكون مفعلة للشبكة، حيث تعتبر غير آمنة. SMTP هي خدمة التوجيه وإعادة الإرسال للبريد، إذا كنا نعمل كمضيف أو خادم بريد، فيفترض أن تكون مفعلة، لكن إذا كنا فقط نقرأ ونكتب رسائل البريد عبر حسابات POP3 و IMAP، فليس من الضروري أن تكون مفعلة.

وهناك طريقة أخرى لاكتشاف الخدمات الفعالة، وتتم عبر البحث عن منافذ الإنصات الفعّالة، ويمكن تنفيذ ذلك بالأمر netstat -lut.

يمكن أيضاً تطبيق الأمر nmap عبر DNS أو عنوان IP للجهاز، يخبرنا هذا بما يبدو عليه النظام من الخارج (في localhost يمكننا أن نرى ما يمكن للجهاز الذي نعمل عليه أن يراه)، أو – الأفضل من ذلك – يمكننا حتى أن نستخدم جهازاً بشبكة خارجية (على سبيل المثال، أي جهاز متصل بالإنترنت) لاختبار ما يمكن رؤيته في جهازنا من الخارج.

سنذهب الآن إلى etc/init.d.conf/ لتعطيل هذه الخدمات. علينا أن ننظر إلى سطور تشبه ما يلي:

```
discard stream tcp nowait root internal
smtp stream tcp nowait mail /usr/sbin/exim exim –bs
```

ونضع الرمز "#" في بداية السطر (فقط للخدمات التي نرغب بتعطيلها، وعندما نعرف ما تفعله هذه الخدمات (تفحص صفحات man لمعرفة إذا كان تعطيلها محبّذاً). وهناك حالة أخرى لما يُنصح بتعطيله، وهي الخدمات ftp و telnet و finger، وعلينا أن نستخدم ssh لاستبدالها.

علينا الآن أن نعيد تشغيل inetd بحيث يقرأ الإعدادات التي قمنا بتغييرها: etc/init.d/inetd restart/.

نعود لـ nmap:

```
22/tcp    open    ssh
80/tcp    open    http
111/tcp   open    sunrpc
111/udp   open    sunrpc
113/tcp   open    auth
631/tcp   open    ipp
728/udp   open    unknown
734/tcp   open    unknown
```

ومما يتبقى، لدينا خدمة ssh التي نرغب بإبقائها تعمل، وخادم الوِب الذي سنوقفه في الوقت الحالي:

```
/etc/init.d/apache stop
```

ipp هي خدمة الطباعة المرتبطة بـ CUPS. رأينا في الجزء التعلق بالإدارة المحلية أنه كانت هناك واجهة وب لـ CUPS يتم الاتصال بها عبر المنفذ 631. إذا كنا نرغب بأخذ فكرة عما يفعله منفذ معين، يمكننا إلقاء نظرة على /etc/services:

```
root@machine:# grep 631 /etc/services
ipp 631/tcp              # Internet Printing Protocol
ipp 631/udp              # Internet Printing Protocol
```

إذا لم نكن نعمل كخادم طباعة لمن هم في الخارج، فعلينا أن نذهب إلى إعدادات CUPS للتخلص من هذه الخاصية (وذلك بوضع listen 127.0.0.1:631، بحيث تستمع للجهاز المحلي فقط)، أو حصر الوصول للأجهزة المصرح لها.

تظهر بعض المنافذ الأخرى كمجهولة، وهي في هذه الحالة 728 و 734؛ يشير هذا إلى أن النظام لم يكن قادراً على تحديد أي nmap يرتبط بالمنفذ. سنحاول رؤيتها بأنفسنا. للقيام بذلك، يمكنا تنفيذ الأمر netstat في النظام، والذي يوفر إحصائيات مختلفة عن أنظمة الشبكة، من الحزم المرسلة والمتلقاة وأخطاء العناصر التي نهتم بها، وهي الاتصالات المفتوحة ومن يستخدمها. سنحاول معرفة من يستخدم المنافذ غير المعروفة:

```
root@machine:~# netstat -anp | grep 728
udp 0 0 0.0.0.0:728 0.0.0.0:* 552/rpc.statd
```

وإذا قمنا بالمثل للمنفذ 734، فيمكننا أن نرى أن rpc.statd هو الذي فتح المنفذ؛ إن rpc.statd مراقب مرتبط بـ NFS (للنظام في هذه الحالة خادم NFS). إذا كررنا هذه العملية على المنفذ 11 – الذي يبدو sunrpc – فسنرى أن المراقب خلف portmap المستخدَم في نظام استدعاء الإجراء البعيد RPC. يسمح النظام RPC للمستخدمين أن يستخدموا

الاستدعاءات البعيدة بين عمليتين على جهازين مختلفين. Portmap مراقب يحول الاستدعاءات التي تصل إلى المنفذ إلى أرقام خدمات RPC الداخلية، وتستخدمه خوادم مختلفة مثل NFS و NIS و +NIS.

يمكن رؤية خدمات RPC المقدمة بالأمر rpcinfo:

```
root@machine:~# rpcinfo -p
program   vers   proto   Port     service
100000    2      tcp     111      portmapper
100000    2      udp     111      portmapper
100024    1      udp     731      status
100024    1      tcp     734      status
391002    1      tcp     39797    sgi_fam
391002    2      tcp     39797    sgi_fam
```

حيث نرى خدمات RPC مع بعض المنافذ التي تم اكتشافها بالفعل. وهناك أمر آخر يمكن أن يكون مفيداً، وهو lsof الذي يجعل من الممكن ربط المنافذ بالخدمات التي قامت بفتحها – إضافة إلى وظائف أخرى – (فمثلاً: lsof -i | grep 731).

المراقب portmap حرج نوعاً ما فيما يتعلق بالأمن، حيث أنه – في الأساس – لا يوفر آليات استيثاق العميل، حيث يفترض أن هذا موكل إلى العملية (NFS أو NIS ...). ونتيجة لذلك، يمكن أن يكون portmap عرضة لهجمات منع الخدمة التي يمكن أن تسبب أخطاء في الخدمات أو توقفها لبعض الوقت. عادة نحمي portmap باستخدام نوع من المغلِّفات wrappers و/أو جدران الحماية. إذا لم نكن نستخدم NFS و NIS ولم نكن نفكر باستخدامها، فإن أفضل ما يمكننا القيام به هو تعطيل portmap كلياً، وإزالته من مستوى الإقلاع المفعّل فيه. يمكننا أيضاً إيقافه مؤقتاً بالنصوص البرمجية التالية (في دبيان):

```
/etc/init.d/nfs-common
/etc/init.d/nfs-kernel-server
/etc/init.d/portmap
```

بإدخال المعامل stop لإيقاف خدمات RPC (وهي في هذه الحالة NFS).

يمكننا بعد ذلك التحكم بالأمن في الأساس باستخدام مغلِّف بسيط. فلنفترض بأننا نرغب بالسماح لجهاز معين بالعبور عبر ssh، ولنقل بأنها تحمل العنوان 1.2.3.4. سنغلق portmap عن الخارج، حيث أنه ليس لدينا NIS، ولدينا خادم NFS

لكننا لا نخدّم شيئاً (يمكننا أن نغلقه، لكننا سنتركه للاستخدام المستقبلي). سننشئ مغلِّفاً (نفترض بأن مغلفات TCP مثبتة مسبقاً) بتعديل الملفات hosts.deny -j allow. في /etc/hosts.deny:

```
ALL : ALL : spawn (/usr/sbin/safe_finger -l @%h \
| /usr/bin/mail -s "%c FAILED ACCESS TO %d!!" root) &
```

نحن نمنع كل الخدمات (كن حذراً، فبعضها مرتبطة بـ inetd)، والخطوة التالية التي سيكون علينا القيام بها هي معرفة من قام بالخدمة ومن أي جهاز، وسنرسل رسالة بريد إلى المستخدم الجذر للإبلاغ عن المحاولة. سنكتب أيضاً ملف تقرير... والآن، في /etc/hosts.allow:

```
sshd 1.2.3.4
```

نحن نسمح بالوصول من الجهاز ذي العنوان 1.2.3.4 إلى خادم sshd (عبر ssh). يمكننا أيضاً إدخال الوصول إلى portmap، وكل ما نحتاجه هو سطر portmap، مثل: la_ip. يمكننا إدراج قائمة بالأجهزة والشبكات الفرعية التي يمكن أن تستخدم الخدمة (انظر إلى man hosts.allow). تذكر أنه لدينا أيضاً الأمر tcpdchk للتأكد من أن إعداد المغلف صحيح، والأمر tcpdmatch لمحاكاة ما يمكن أن يحدث في محاولة معينة، على سبيل المثال:

```
root@machine:˜# tcpdmatch sshd 1.2.3.4
warning: sshd: no such process name in /etc/inetd.conf client: hostname
machine.domain.es
client: address 1.2.3.4
server: process sshd
matched: /etc/hosts.allow line 13
access: grantedv
```

يخبرنا بأنه سيتم تقديم وصول. ومن بين التفاصيل، يخبرنا بأن sshd ليس في inetd.conf، وإذا تحققنا منه فسنرى بأنه غير موجود هناك: لا يتم تفعيله من خادم inetd، بل من المراقب في مستوى التشغيل الذي نعمل عليه. إضافة إلى ذلك، فهو في دبيان مصرّف بمكتبة مغلّف مدمجة فيه (وبالتالي لا يحتاج لأن يكون tcpd عاملاً). في دبيان، هناك العديد من المراقبات، مثل: ssh, portmap, in.talk, rpc.statd, rpc.mountd، إضافة إلى غيرها. يسمح لنا هذا بتأمين هذه المراقبات باستخدام مغلفات.

وهناك أمر آخر يفترض أن يتم التحقق منه ويتعلق بالاتصالات الحالية الموجودة. يمكننا بالأمر netstat -utp أن

نعرض قائمة باتصالات tcp و udp المنشأة خارجياً، سواء كانت واردة أم صادرة؛ ولهذا، يمكننا – في أي وقت – أن نكتشف العملاء المتصلين والذين نتصل معهم. وهناك أمر آخر هام (بوظائف مختلفة) وهو lsof الذي يمكنه ربط الملفات المفتوحة بالعمليات أو الاتصالات على الشبكة عبر lsof -i، مما يسمح بمعرفة أيّ عمليات وصول غير مناسبة إلى الملفات.

يمكننا أيضاً أن نستخدم جدار حماية للعمليات المشابهة (أو كآلية آضافية). سنبدأ برؤية وضع قواعد الجدار الناري في ذلك الوقت: (الأمر iptables -L)

```
root@aopcjj:~# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source                    destination
Chain FORWARD (policy ACCEPT)
target prot opt source                    destination
Chain OUTPUT (policy ACCEPT)
target prot opt source                    destination
```

وبعبارة أخرى، لا يضع جدار الحماية أيّة قيود في هذا الوقت، ويمكن إرسال واستقبال وإعادة إرسال كل الحزم.

عند هذه النقطة، يمكننا أن نضيف جدار حماية يقدم إدارة أفضل للحزم التي نستقبلها ونرسلها، والتي يمكن أن تكون تحكماً أولياً لتحسين الأمن. يمكننا أن نبني القواعد الضرورية بشكل مشابه لأمثلة جدار الحماية المقدمة في هذه الوحدة حسب احتياجاتنا.

عند ضبطنا لجدران الحماية، فيمكننا أن نختار استخدام هذه الآلية كعنصر أمني وحيد وإزالة المغلّفات: يمكن عمل هذا، وذلك لأن جدران الحماية تقدم (في هذه الحالة عبر iptables) ميزة قوية جدًّا تسمح لنا بتتبع حزمة حسب نوعها أو ميفاقها أو ما تفعله في النظام. يمكن أن يكون جدار حماية جيد كافياً، لكن لنكون آمنين، ستكون الإجراءات الأمنية الإضافية مفيدة دائماً. وإذا لم يكن جدار الحماية مصمماً جيداً، وكان يترك بعض الحزم تفلت منه، فسيكون المغلِّفُ الإجراءَ الذي يعمل على مستوى الخدمة لإيقاف أيّ وصول غير مرغوب فيه. ولنتخذ هذا المجاز شائع الاستخدام كمثال، إذا كنا نعتبر نظامنا قلعة من العصور الوسطى يجب حمايتها، فسيكون الخندق المائي والجدار الأمامي جدار الحماية، وسيكون جدارُ العزل الثاني المغلِّف.

الأنشطة

١) لنفترض بأنه لدينا موقع في جهازنا، باستخدام أباتشي مثلاً. موقعنا مصمم لعشرة مستخدمين محليين، ولكننا لا نتحكم بهذا العدد. ونتيجة لذلك، نفكر بجعل هذا النظام متاحاً على الإنترنت، حيث نعتبر أنه سيكون مفيداً للعملاء، والشيء الوحيد الذي علينا فعله هو إعطاء النظام عنوان IP عام على الإنترنت. ما أنواع الهجمات التي يمكن أن يعاني منها هذا النظام؟

٢) كيف يمكننا اكتشاف الملفات عبر suid في نظامنا؟ ما الأوامر الضرورية؟ والمجلدات ذات SUID و SGID ؟ ولم من الضروري أن يكون للملف etc/passwd/ مثلاً خانة SUID ؟

٣) إن ملفات rhosts. – كما رأينا – خطر كبير على النظام. هل يمكننا استخدام نوع من الأساليب الآلية لفحص وجود هذه الملفات دورياً؟ وكيف يكون ذلك؟

٤) لنفترض أننا نرغب بتعطيل خدمة نعلم بأن لها نصاً برمجياً etc/init.d/service/ يتحكم بها: نرغب بتعطيلها في كل مستويات التشغيل التي تظهر فيها. كيف نجد مستويات التشغيل الموجودة فيها؟ (على سبيل المثال، بالبحث عن روابط للنص البرمجي).

٥) اختبر الخدمات الفعالة في جهازك. هل كلها ضرورية؟ كيف نحميها أو نعطلها؟

٦) تدرب على استخدام بعض الأدوات الأمنية المذكورة (nmap، nessus، إلخ).

٧) ما قواعد Iptables الضرورية لجهاز نرغب بالوصول إليه فقط عبر SSH من عنوان معين؟

٨) ماذا إذا كنا نرغب بالوصول فقط إلى خادم البريد؟

المراجع

مصادر أخرى للمراجع والمعلومات:

[Hatc] [Deb] المواقع الأمنية للتوزيعات.

[Peñ] ضروري لدبيان، مع وصف جيد جداً لكيفية ضبط الأمن، ويمكن تتبعه خطوة بخطوة، و [Hatb] هو ما يكافئه لفيدورا/ردهات.

[Mou01] مرجع أمني ممتاز لردهات (ينطبق على دبيان أيضاً).

[Hat01] كتب أمنية لجنو/لينكس تغطي النواحي والتقنيات بشكل مفصل.

[Line] دليل صغير (من صفحتين) للأمن.

[Sei] دليل خطوة بخطوة يحدد النقاط الأساسية التي سيكون علينا التحقق منها والمشكلات التي يمكن أن تطرأ.

[Net] المشروع Netfilter و IPtables

[Ian] قائمة بمنافذ TCP/IP

[Nes] [Insb] [Sno] [Proa] بعض الأدوات الأمنية الأكثر استخداماً

[NSAb] إصدار لينكس موجه للأمن، تصدره وكالة الأمن القومي الأمريكية NSA. مرجع لـ SELinux.

[NSAa] [Incb][Insa][Aus][CERa] مواقع المنظمات الأمنية.

[San][Ins][CERb] ثغرات واستغلالات لأنظمة التشغيل المختلفة.

[USA][FBI][NSAa] بعض "سياسات" الجريمة الإلكترونية في الولايات المتحدة الأمريكية.

مراجع عربية:

كتاب *SELinux Arabic Guide* للمهندس صبري صالح. <http://king-sabri.net/files/SELinux-ar.pdf>

المواضيع في قسم أمن الشبكات والأنظمة في مجتمع لينكس العربي http://www.linuxac.org/forum/forums/4

تضبيط الإعدادات وتحسين الأداء

د. رِمو سُـپي بُلدريتو

مقدمة

من النواحي الأساسية – بمجرد ضبط النظام – إعداد وضبط النظام بحيث يتوافق مع احتياجات المستخدم للتأكد من أن المزايا مطابقة قدر الإمكان للاحتياجات التي ستطبق عليها. إن جنو/لينكس نظام كفؤ يقدم مستوىً ممتازاً من الإعدادات الممكنة وكثير من التحسينات المحببة التي يمكن أن يتم تفصيلها حسب احتياجات المستخدم. وبهذه الطريقة، بمجرد تثبيت النظام (أو تحديثه، اعتماداً على الحالة)، فهناك إعدادات معينة ضرورية للنظام يجب تضبيطها. رغم أن النظام قد "يعمل"، فمن الضروري القيام ببعض التغييرات (تطويع البيئة أو تضبيطها)، بحيث تتم مطابقة كل احتياجات المستخدمين والخدمات التي يجب أن يقدمها الجهاز. سيعتمد هذا التضبيط على مكان عمل الجهاز؛ سيتم القيام بالتضبيطات – في بعض الحالات – من أجل تحسين أداء وكفاءة النظام. و إضافة إلى ذلك، ففي حالات أخرى يكون ذلك لأسباب أمنية (انظر إلى الجزء التاسع "إدارة الأمن"). عندما يكون النظام عاملاً، من الضروري مراقبة النظام ورؤية كيف يعمل أو يتصرف، والتصرف بناء على ذلك. رغم أن هذه ناحية أساسية، فعادة يتم ترك تضبيط نظام التشغيل لخبراء الحاسوب؛ لكن إذا كنا نعرف المعاملات التي تؤثر على الأداء، فمن الممكن تحقيق حلول جيدة بالقيام بعمليات تحليل دورية، وعمل تغييرات على الإعداد، والمراقبة، وعمل تضبيطات.

1 نواحٍ أساسيّة

قبل التعرف على تقنيات التحسين، من المهم عمل قائمة بالأسباب التي يمكن أن تؤثر على أداء نظام التشغيل. ومن بينها، يمكنا أن نذكر ما يلي:

1) ظاهرة عنق الزجاجة في الموارد[1]: نتيجة لذلك سيكون النظام بأكمله أبطأ وذلك بسبب وجود موارد لا يمكنها تلبية الاحتياجات المطلوبة منها. الخطوة الأولى لتحسين النظام هي إيجاد أعناق الزجاجات هذه ومسبباتها، بينما نتعرف على حدودها النظرية والعملية.

2) قانون أمداهل Amdahl، طبقاً لهذا القانون، "هناك حدّ للقدر الذي يمكن به تحسين (أو تسريع) النظام ككل عند تحسين جزء واحد من النظام فقط"، وبعبارة أخرى، إذا كان لدينا تطبيق يستخدم 10% من المعالج، وقد تم تحسينه لتقليل هذا الاستخدام بمعامل قدره 2، فسيتحسن أداء (سرعة) البرنامج بمقدار 5%، مما يعني أنه قد تم صرف جهد كبير جداً لعمل شيء لا يعادل النتائج التي نسعى إليها.

3) تقديرات التسريع: من الضروري تقدير المقدار الذي سيتحسن فيه النظام من أجل تجنب أي مجهود أو كلفة غير ضروريين. يمكننا استخدام القانون المذكور آنفاً لتقييم ما إذا كان من الضروري استثمار الوقت أو المال في النظام.

4) ظاهرة الفقاعة: من الشائع جداً أن يكون هناك شعور بأنه بمجرد حلنا لمشكلة ما فستظهر دائماً مشكلة أخرى. ومن أعراض هذه المشكلة أن النظام يتنقل باستمرار بين مشاكل المعالج ومشاكل الدخل/الخرج، والعكس.

---

1 عند تعبئة زجاجة بالماء ثم محاولة إفراغها، ستعتمد سرعة خروج الماء على الاتساع الأقل، فقطر الزجاجة نفسها يمكنه تمرير كمية معينة من الماء خلال مدة زمنية معينة، لكن لا يمكن لعنق الزجاجة مجاراة تلك السرعة لأنه أقلّ اتساعاً، وبهذا تعتمد سرعتنا في إفراغها على الأقلّ اتساعاً، وهو عنق الزجاجة، وقد بنيت على هذه الفكرة نظريات عديدة. إذا كان لدينا قرصان صلبان بسرعتين مختلفتين وأردنا النسخ من أحدهما إلى الآخر فستعتمد سرعة النسخ على السرعة الأقل. وإذا كان لدينا قرصان سرعة القراءة من كل منهما س، وكانا موصولين على منفذ سرعته 1.5س، فستكون أكبر سرعة للقراءة من كليهما معاً 1.5 س، وليس 2س، وذلك لأن المنفذ يشكل عنق الزجاجة في هذه الحالة؛ ولهذه الظاهرة تطبيقات في تقنيات RAID على سبيل المثال.

5) وقت الاستجابة مقارنة بضغط العمل: إذا كان لدينا عشرون مستخدماً، فإن تحسين الإنتاجية سيعني أن يحصل الجميع على إنجازات أكثر في نفس الوقت، لكن وقت الاستجابة الوحيد لن يتحسن؛ قد يكون سبب ذلك هو أن أوقات الاستجابة للبعض قد يكون أفضل مما هي عليه لغيرهم. تحسين أوقات الاستجابة يعني تحسين النظام بحيث تأخذ المهام المنفردة وقتاً قليلاً قدر الإمكان.

6) علم نفس المستخدم: هناك معاملان هامّان: 1) عامة لن يكون المستخدم راضياً عند وجود اختلافات في وقت الاستجابة؛ 2) لن يلاحظ المستخدم أيّ تحسينات في وقت التنفيذ تقل عن 20%.

7) أثر الاختبار: قياسات المراقبة تؤثر على القياسات نفسها. علينا الاستمرار بحذر عند قيامنا باختبارات، بسبب الأثر الناتج عن برامج الاختبار نفسها.

8) أهمية المعدل والاختلافات: علينا أخذ النتائج بعين الاعتبار، علماً بأننا إذا حصلنا على معدل استخدام للمعالج قدره 50% عند استخدام 100 , 0 , 0 , 100، فقد نخرج بنتائج خاطئة. فعلينا أن نرى الاختلافات في المعدل.

9) معرفة أساسية بعتاد الجهاز الذي سيتم تحسينه: لتحسين شيء ما، علينا "معرفة" ما إذا كان من الممكن تحسينه. يجب أن يكون لدى الشخص المسؤول عن التحسين معرفة أساسية كبيرة عن العتاد المستخدَم (المعالج، الذاكرة، المنافذ، الكاش، الدخل/الخرج، الأقراص، المرئيات، ...) والاتصالات المتبادلة من أجل تحديد مكان وجود المشكلة.

10) معرفة أساسية بنظام التشغيل المراد تحسينه: كما في النقطة السابقة، يجب على المستخدم أن يعلم الحد الأدنى من النواحي المتعلقة بنظام التشغيل الذي يرغب بتحسينه، والتي قد تتضمن مفاهيم كالعمليات، وخيوط المعالجة (الإنشاء، والتنفيذ، والحالات، والأولويات، والإنهاء)، واستدعاءات النظام، وتمرير البيانات إلى ذاكرة كاش cache buffers، ونظام الملفات، وإدارة الذاكرة، والذاكرة الحيوية (ترحيل الصفحات والإبدال)، وجداول النواة.

1.1 المراقبة في نظام يونكس System

سيظهر /proc/ كمجلد، ولكنه في الحقيقة نظام ملفات وهمي، وبعبارة أخرى، هو غير موجود على القرص، وتنشئه النواة في الذاكرة. ويستخدم لتقديم معلومات عن النظام (وبالأصل كان لعرض معلومات العمليات، كما هو واضح في الاسم)، والذي سنستخدمه في الأوامر التي سنبحثها الآن. سننظر الآن إلى بعض الملفات المثيرة للاهتمام (تفقد الصفحة ذات العلاقة في دليل الاستخدام man لمزيد من المعلومات):

/proc/1: مجلد يحوي معلومات العملية الأولى (إن رقم المجلد يشير إلى رقم العملية PID في النظام)

/proc/cpuinfo: معلومات عن المعالج (النوع، الصنف، الموديل، الأداء، ...).

/proc/devices: قائمة بالأجهزة المضبوطة في النواة.

/proc/dma: قنوات DMA المستخدمة في هذه اللحظة.

Proc/filesystems/: أنظمة الملفات المضبوطة في النواة.

Xproc/interrupts/: تعرض التقاطعات المستخدَمة وما تم معالجته منها.

/proc/ioports: ينطبق نفس الأمر على المنافذ.

Xproc/kcore/: صورة الذاكرة الفيزيائية للنظام.

Xproc/kmsg/: الرسائل التي أنشأتها النواة والتي يتم بعد ذلك إرسالها إلى syslog.

Xproc/ksyms/: جدول برموز النواة.

Xproc/loadavg/: حمل النظام.

Xproc/memingo/: معلومات عن استخدام الذاكرة.

Xproc/modules/: وحدات حمّلتها النواة.

Xproc/net/: معلومات عن موافيق الشبكة.

Xproc/stat/: إحصائيات عن النظام.

Xproc/uptime/: منذ متى يعمل النظام.

Xproc/version/: إصدار النواة.

علينا أن نتذكر أن هذه الملفات مرئية (نصوص) لكن البيانات أحياناً قد تكون في هيئة "خام"، ومن الضروري استخدام أوامر لتفسيرها. تلك الأوامر هي ما سنبحثه الآن.

تستخدم الأنظمة المتوافقة مع يونكس SV الأمرين sar و sadc للحصول على إحصائيات النظام (مضمنة في فيدورا ضمن الحزمة sysstat التي تتضمن أيضاً iostat و mpstat). المكافئ لها في جنو/لينكس دبيان هو atsar (و atsadc)، وهو المكافئ نفسه لما ذكرنا. يقرأ الأمر atsar العدادات والإحصائيات التي في الملف /proc/، وتظهرها في الخرج القياسي. الطريقة الأولى لاستدعاء الأمر هي:

```
atsar options t [n] n
```

حيث يُعرض النشاط لعدد n من المرات كل t ثانية بترويسة تظهر عداد النشاط (القيمة المبدئية لـ n هي 1). الطريقة الثانية لاستدعائه هي:

```
atsar -options -s time -e time -i sec -f file -n day
```

يستخرج الأمر البيانات من الملف المحدد في f- (وهو مبدئياً الملف /var/log/atsar/atsarxx، مع كون xx اليوم في الشهر) والذي حفظناه سابقاً بالأمر atsadc (الذي يُستخدَم لجمع المعلومات وحفظها ومعالجتها، و في دبيان يكون في /usr/lib/atsar). يمكن استخدام المعامل n للإشارة إلى اليوم في الشهر، و s- و e- الوقت الذي تم فيه آخر إقلاع، على الترتيب.لتفعيل atsadc على سبيل المثال، يمكننا أن نضمِّن سطراً في /etc/cron.d/atsar كما يلي:

```
@reboot root test -x /usr/lib/atsadc && /usr/lib/atsar/atsadc /var/log/atsar/atsa'date +\%d'
10,20,30,40,50 * * * * root test -x /usr/lib/atsar/atsa1 && /usr/lib/atsar/atsa1
```

يُنشئ الأول الملف بعد إعادة التشغيل. والثاني يحفظ البيانات كل 10 دقائق بالنص البرمجي atsa1 الذي يستدعي atsadc.

في atsar (أو sar)، الخيارات المستخدمة للإشارة إلى أي الخيارات يجب عرضها، تتضمن بعض الأمثلة ما يلي:

الوصف	الخيار
استخدام المعالج	u
النشاط على الأقراص	d
عدد التقاطعات	I (حرف i)
استخدام الجداول في النواة	V
إحصائيات استخدام الطرفيات tty	and
معلومات عن نشاط ترحيل الصفحات paging والإبدال swap	P
الذاكرة الحرة واستخدام swap	r
إحصاءات الشبكة	l (حرف L)
معلومات أخطاء الشبكة	L
إحصاءات اتصال IP	w
إحصاءات TCP	t
إحصاءات UDP	U
إحصاءات ICMP	m
إحصاءات NFS	N
كل الخيارات	A

ملاحظة

المراقبة باستخدام atsar:

- المعالج: atsar -u
- الذاكرة: atsar -r
- القرص: atsar -d
- ترحيل الصفحات: atsar -p

هناك فقط اختلافات بسيطة بين atsar و sar في طريقة عرض البيانات، ويتضمن sar بعض الخيارات الإضافية (أو المختلفة). سنرى الآن بعض الأمثلة على كيفية استخدام sar (كما في atsar تماماً، والاختلاف الوحيد هو في كيفية

عرض البيانات) ومعاني المعلومات التي تنشئها:

1) استخدام المعالج:

```
sar -u 4 5
Linux 2.6.19-prep (localhost.localdomain)  24/03/07
08:23:22    CPU   %user   %nice   %system   %iowait   %steal   %idle
08:23:26    all   0.25    0.00    0.50      0.00      0.00     99.25
08:23:30    all   0.00    0.00    0.00      0.00      0.00     100.00
08:23:34    all   0.00    0.00    0.00      0.00      0.00     100.00
08:23:38    all   0.25    0.00    0.00      0.00      0.00     99.75
08:23:42    all   0.00    0.00    0.00      0.00      0.00     100.00
Media:      all   0.10    0.00    0.10      0.00      0.00     99.80
```

- تعرض user و system نسبة استخدام وقت المعالج في وضع المستخدم بقيمة nice = 0 (الوضع الاعتيادي) ووضع النواة.

- Nice هو نفسه لكن مع عمليات المستخدم nice < 0 (أقل من الأولوية المتوسطة).

- يشير idle إلى الوقت الذي كان فيه المعالج حرًّا عندما كان النظام يدخُل أو يخرج من قرص.

- Steal هو الوقت الذي تم تضييعه دون فائدة أثناء انتظار معالج افتراضي (مطبق في بيئات المحاكاة).

في هذه الحالة idle=100، مما يعني أن المعالج خامل، مما يعني أنه لا توجد عمليات للتنفيذ، وأن الحمل على الجهاز ضئيل؛ إذا كان idle=10 و كان هناك عدد كبير من العمليات، فيجب أن يؤخذ تحسين المعالج بعين الاعتبار، حيث يمكن أن يكون هناك عنق زجاجة في النظام.

2) عدد التقاطعات في الثانية

```
sar -I 4 5
Linux 2.6.19-prep (localhost.localdomain)        24/03/07
08:24:01 INTR intr/s
08:24:06 4 0.00
Media: 4 0.00
```

تعرض المعلومات الموجودة في proc/interruprs/ عن مدى استخدام التقاطعات للمستويات الفعالة. هذا مفيد لمعرفة إذا كانت هناك أجهزة تقاطع عمل المعالج باستمرار.

3) الذاكرة والإبدال

```
sar -r 4 5
Linux 2.6.19-prep (localhost.localdomain)        24/03/07
 08:24:20 kbmemfree kbmemused %memused kbbuffers kbcached kbswpfree
kbswpused %swpused kbswpcad
08:24:24 296516 729700 71.11 24260 459972 963860 0 0.00 0
08:24:28 296500 729716 71.11 24268 459968 963860 0 0.00 0
08:24:32 296516 729700 71.11 24268 459976 963860 0 0.00 0
08:24:36 296516 729700 71.11 24276 459976 963860 0 0.00 0
08:24:40 296500 729716 71.11 24276 459976 963860 0 0.00 0
Media: 296510 729706 71.11 24270 459974 963860 0 0.00 0
```

في هذه الحالة kbmemfree هي الذاكرة الحرة الرئيسية (MP)، used هي الذاكرة المستخدمة، و buffers هي (MP) المستخدمة في buffers، cache هي الذاكرة الرئيسية المستخدمة في الحفظ المؤقت للصفحات، swpfree/used هي مساحة الإبدال الحرّة/المستخدمة. من الضروري تذكر أنه إذا لم تكن هناك مساحة في MP، فسينتهي المطاف بصفحات العمليات في ذاكرة الإبدال، حيث يفترض أن تكون هناك مساحة. يفترض أن يتم مقارنة هذا باستخدام المعالج. علينا أن نتأكد أيضاً أن حجم الـ buffers مناسب وأنه كذلك مقارنة بالعمليات التي تقوم بعمليات إدخال/إخراج (I/O).

من المفيد أيضاً تجربة الأمر free، حيث يسمح لنا برؤية مقدار الذاكرة بطريقة مبسطة:

	total	used	free	shared	buffers	cached
Mem:	1026216	729716	296500	0	24324	459980
-/+ buffers/cache:	245412	780804				
Swap:	963860	0	963860			

يشير هذا إلى أن ¾ الذاكرة التي قيمتها 1 جيجا مستخدمة، إضافة إلى حوالي ½ الكاش أيضاً. يخبرنا هذا أيضاً أنه لا

يتم استخدام الإبدال في الوقت الحالي لأي شيء، مما يعني أنه يمكننا استنتاج أن النظام في حالة جيدة. إذا كنا نرغب برؤية المزيد من التفاصيل، فعلينا أن نرى الأمر vmstat (أو sar -r) لتحليل ما يسبب المشكلة أو ما يستهلك ذاك القدر من الذاكرة. وفيما يلي مخرجات vmstat 1 10:

procs	-----------memory----------	---swap--		-----io----		--system--			-----cpu------
r b	swpd free buff cache si so bi	bo	in	cs	us	sy	id	wa	st
0	0 0 295896 24384 459984 0 0 321	56	1249	724	11	2	81	5	0
0	0 0 295896 24384 459984 0 0 0	28	1179	383	1	0	99	0	0
0	0 0 295896 24384 460012 0 0 0	0	1260	498	0	0	100	0	0
0	0 0 295896 24384 460012 0 0 0	0	1175	342	0	0	100		0
0	0 0 295896 24384 460012 0 0 0	0	1275	526	0	0	100	0	0
1	0 0 295896 24392 460004 0 0 0	72	1176	356	0	0	99	1	0
0	0 0 295896 24392 460012 0 0 0	0	1218	420	0	0	100	0	0
0	0 0 295896 24392 460012 0 0 0	0	1216	436	0	0	100	0	0
0	0 0 295896 24392 460012 0 0 0	0	1174	361	0	0	100	0	0
1	0 0 295896 24392 460012 0 0 0	0	1260	492	0	0	100	0	0

4) استخدام جداول النواة

Linux 2.6.19-prep (localhost.localdomain) 24/03/07

08:24:48	dentunusd	file-sz	inode-sz	super-sz	%super-sz	dquot-sz	%dquot-sz	rtsig-sz	%rtsig-sz
08:24:52	19177	3904	15153	0	0.00	0	0.00	0	0.00
08:24:56	19177	3904	15153	0	0.00	0	0.00	0	0.00
08:25:00	19177	3904	15153	0	0.00	0	0.00	0	0.00
08:25:04	19177	3904	15153	0	0.00	0	0.00	0	0.00
08:25:08	19177	3904	15153	0	0.00	0	0.00	0	0.00
Media:	19177	3904	15153	0	0.00	0	0.00	0	0.00

في هذه الحالة، superb-sz هو أكبر عدد من الـ superblocks التي تتصرف فيها النواة في الوقت الحالي لأنظمة الملفات المضمومة؛ inode-sz هو العدد الأقصى لـ incore-inode الضرورية للنواة في الوقت الحالي، وهي واحدة على الأقل لكل قرص؛ file-sz هو العدد الأقصى للملفات المفتوحة؛ dquota-sz هو أكبر قدر مستخدم من حصة الدخل (للخيارات المتبقية، راجع صفحات man لـ sar (أو atsar)). يمكن إكمال عملية المراقبة هذه بالأمر ps -edaflm (حالة العمليات)، والأمر top، والذي سيعرض نشاط وحالة عمليات النظام. ما يلي مثالان على كلي الأمرين (فقط بعض السطور):

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	STIME	TTY	TIME	CMD	
4	-	root	1	0	0	-	-	-	508	-	08:01	?	00:00:00	init [5]	
1	-	root	1927	7	0	-	-	-	0	-	08:02	?	00:00:00	[kondemand/0]	
1	-	rpc	2523	1	0	-	-	-	424	-	08:02	?	00:00:00	syslogd -m 0	
5	S	rpc	2566	1	0	-	-	-	444	-	08:02	?	00:00:00	portmap	
5	-	root	-	0	78	0	-	-	-	-	08:02	-	00:00:00	-	
5		root	2587	1	0	-	-	-	472	-	08:02	?	00:00:00	rpc.statd	
5	S	root	-	-	0	81		0	-	-	08:02	-	00:00:00	-	
1	-	root	2620		1	0	-	-	1232	-	08:02	?	00:00:00	rpc.idmapd	
1	S	root	-	-	0	75		0	-	defaul	08:02	-	00:00:00	-	
5	-	root	2804	1	0	-	-	-	1294	-	08:02	?	00:00:00	/usr/sbin/sshd	
5	S	root	-	-	0	84	0	-	-	-	08:02	-	00:00:00	-	
5	-	root	2910	1	0	-	-	-	551	-	08:02	?	00:00:00	/usr/sbin/atd	
5	S	root	-	-	0	84	0	-	-	-	08:02	-	00:00:00	-	
4	-	root	3066	1	0	-	-	-	407	-	08:02	tty1	00:00:00	/sbin/mingetty tty1	
4		root	3305	1	0	-	-	-	21636	-	08:03	?	00:00:01	nautilus --no-default-	
window --sm-															
4	-	root	3305	1	0	-	-	-	21636	-	08:03	?	00:00:01	client-id default3	
0	-	root	3643	3541	0	-	-	-	1123	-	08:17	pts/1		00:00:00	bash
4	-	root	3701	3643	0	-	-	-	1054	-	08:27	pts/1		00:00:00	ps -edaflm

…

حيث تعكس المعاملات القيم المشار إليها في متغير النواة لهذه العملية، وأكثرها أهمية من وجهة نظر المراقَبة هي: أوسمة F (يشير الرقم 1 في هذه الحالة إلى الصلاحيات القصوى، ويشير 4 إلى أنه أنشأها مراقب البدء)، و S هي الحالة (D: دخل/خرج نائم لا يمكن مقاطعته، و R: يمكن تشغيله أو طابور تشغيل، S: نائمة، T: يتم تتبعها أو متوقفة، Z: عملية معطوبة (حيّة ميتة أو زومبي Zombie). و PRI هي الأولوية، NI تعني nice، و STIME هي وقت بدء التنفيذ، TTY الطرفية التي تم تنفيذها فيها، TIME وقت المعالج، CMD هو البرنامج الذي شغّل معاملاتها. إذا كنا نرغب بتحديث الصفحة (يمكن ضبط ذلك)، فيمكننا استخدام الأمر top، الذي يعرض الإحصاءات العامة (العمليات، الحالات، الحمل، إلخ) ومن ثم يحصل على معلومات عن كل منها، بما يشبه ps، لكن يتم تحديثها مبدئياً كل 5 ثوان.

```
top - 08:26:52 up 25 min, 2 users, load average: 0.21, 0.25, 0.33
Tasks: 124 total, 1 running, 123 sleeping, 0 stopped, 0 zombie
Cpu(s): 10.8%us, 2.1%sy, 0.0%ni, 82.0%id, 4.9%wa, 0.1%hi, 0.1%si, 0.0%st
Mem: 1026216k total, 731056k used, 295160k free, 24464k
buffers
 Swap: 963860k total, 0k used, 963860k free, 460208k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3541	root	15	0	42148	14m	981	S	1.9	1.5	0:00.76	gnome-terminal
3695	root	15	0	260	944	1650	R	1.9	0.1	0:00.02	top
1	root	RT	0	2032	680	580	S	0.0	0.1	0:00.85	init
2	root	34	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	RT	19	0	0	0	S	0.0	0.0	0:00.04	ksoftirqd/0
4	root	10	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	16	-5	0	0	0	S	0.0	0.0	0:00.00	events/0
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
7	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
53	root	11	-5	0	0	0	S	0.0	0.0	0:00.01	kblockd/0
54	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
177	root	18	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
178	root	18	-5	0	0	0	S	0.0	0.0	0:00.00	ksuspend_usbd
181	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khubd
183	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	kseriod
203	root	23	0	0	0	0	S	0.0	0.0	0:00.00	pdflush
204	root	15	0	0	0	0	S	0.0	0.0	0:00.03	pdflush

يحوي دبيان لينكس أيضاً مجموعة كاملة من أدوات المراقبةالمكافئة لـ sar، ولكن تم إنشاؤها في يونكس BSD والتي لها وظائف مشابهة، ولكن من أوامر مختلفة. vmstat (إحصاءات المعالج والذاكرة والدخل/الخرج)، و iostat (إحصاءات الأقراص والمعالج)، و uptime (حمل النظام والحالة العامة).

1.2 تحسين النظام

سنلقي نظرة الآن على بعض النصائح المتعلقة بتحسين النظام والتي تتعلق بالبيانات التي يتم الحصول عليها.

1) معالجة مشاكل الذاكرة الرئيسية

علينا التأكد من أن الذاكرة الرئيسية بإمكانها التعامل مع نسبة كبيرة من العمليات المنفّذة، وإلّا فقد يقوم النظام بترحيل الصفحات وإحالتها إلى الإبدال؛ لكن هذا يعني أن تنفيذ تلك العملية سيتدهور بشكل كبير. إذا أضفنا المزيد من الذاكرة، فسيتحسن وقت الاستجابة بشكل كبير. ولعمل هذا، يجب أن نأخذ بعين الاعتبار حجم العمليات (SIZE) والحالة R وإضافة ما تستخدمه النواة – وهو ما يمكن الحصول عليه بالأمر dmesg – الذي سيعرض لنا ما يلي على سبيل المثال (أو بالأمر free):

```
Memory:
255048k/262080k available (1423k kernel core, 6644k reserved, 466k data, 240k
init, Ok highmem
```

يجب علينا بعد ذلك أن نقارن بين هذا والذاكرة الفيزيائية وتحليل ما إذا كان النظام مقيداً بالذاكرة (يمكننا أن نرى كثيراً من أنشطة الترحيل بالأمر atsar -r و p-).

حلول مشاكل الذاكرة بديهية: إما أن نزيد السعة، أو نقلل الطلب عليها. اعتماداً على السعر الحالي للذواكر، فزيادة السعة أفضل من قضاء ساعات طويلة في محاولة لتحرير بضع مئات من البايتات فقط، عبر حذف، وإزالة، وترتيب، وتقليل متطلبات العمليات التي تعمل. يمكن تقليل المتطلبات بتقليل جداول النواة، وحذف الوحدات، وتحديد حدّ أقصى لعدد المستخدمين، وتقليل الـ buffers، إلخ، وكل ذلك سيقلل من كفاءة النظام (ظاهرة الفقاعة) وسيكون الأداء أسوأ (وفي بعض الحالات سيصير النظام غير قابل للاستخدام مطلقاً).

من النواحي الأخرى التي يمكن تقليلها هي مقدار الذاكرة للمستخدمين، وأزالة أيّة عمليات مكررة، وتغيير الحمل. ومن أجل عمل ذلك، يجب علينا مراقبة العمليات المعطوبة (زومبي) والتخلص منها، وتلك التي لا تتعامل مع الدخل/الخرج (بمعرفة ما إذا كانت عمليات نشطة، ومقدار المعالجة التي تستخدمها، وإذا كان المستخدمون يريدونها). تغيير الحمل هو استخدام تخطيط

الطابور، بحيث يصير بإمكان العمليات التي تحتاج مقداراً كبيراً من الذاكرة أن تعمل عندما يكون هناك نشاط قليل (في الليل – على سبيل المثال – باستخدام الأمر at لتشغيلها).

2) استخدام كبير جدًّا للمعالج

بشكل أساسي، يمكننا الحصول على ذلك من وقت الحمول (قيم منخفضة). علينا أن نحلل – عبر ps أو top – لنعرف أي العمليات "تلتهم" المعالج، واتخاذ قرارات مثل: تأجيل تنفيذها، أو إيقافها مؤقتاً، أو تغيير أولويتها (الأقل تعارضاً بينها كلها، ويتم بأمر إعادة إعطاء الأولوية)، أو تحسين البرنامج (للمرة المقبلة)، أو تغيير المعالج (إضافة واحد آخر). وكما ذكرنا، يستخدم جنو/لينكس المجلد /proc/ لإبقاء كل متغيرات إعداد النواة، والتي يمكن تحليلها، و تضبيطها – في حالات بعينها – لتحقيق مستويات أداء مختلفة أو محسنة.

لعمل ذلك، يجب علينا أن نستخدم الأمر systune dump > /tmp/sysfile للحصول على كل المتغيرات وقيمها في الملف /tmp/sysfile (وفي توزيعات أخرى، يكن عمل ذلك بالأمر sysctl). يمكن تعديل هذا الملف، بتغيير المعاملات ذات العلاقة، ومن ثم استخدام الأمر systune -c /tmp/sysfile لتحميلها في /proc/. يقرأ الأمر systune مبدئياً أيضاً إذا لم يكن لدينا الخيار c- في /etc/systune.conf. في هذه الحالة – على سبيل المثال – يمكننا تعديل (استمر بحذر، حيث يمكن أن يخرب هذا النواة) المتغيرات من النوع /proc/sys/vm (الذاكرة الافتراضية) أو /proc/sys/kernel (ضبط قلب النواة).

وبنفس الطريقة، من الممكن أيضاً (للخبراء، أو من ليس لديهم شيء ليخسرونه) تغيير الحدّ الأقصى لوقت الشريحة – التي يوكلها مجدول النظام إلى كل عملية بطريقة دورية (ينصح باستخدام renice عملياً). لكن جنو/لينكس – على النقيض من أنظمة التشغيل الأخرى – له قيم ثابتة في الكود، حيث يتم تحسينه لوظائف مختلفة (لكن من الممكن تعديل ذلك). يمكننا أن "نعبث" (على مسؤوليتنا الخاصة) بمجموعة من المتغيرات التي يتجعل من الممكن التعامل مع شريحة الوقت time slice الموكلة إلى المعالج (الملف kernel/sched.c في المصدر البرمجي للنواة).

3) تقليل عدد الاستدعاءات

وهناك طريقة أخرى عملية لتحسين الأداء ألا وهي تقليل عدد استدعاءات النظام التي تستهلك معظم وقت المعالج. هذه الاستدعاءات هي عادة ما تستحضره ()fork و ()exec للطرفية. إن الضبط غير الملائم لمتغير النظام PATH، ونتيجة لكون ()exec لا تحفظ أي شيء في ذاكرة كاش، فقد يكون للمجلد الحالي (المشار إليه بالرمز ./) علاقة تنفيذ سلبية. ونتيجة لذلك، علينا دائماً أن نضبط المتغير PATH للمجلد الحالي كآخر مسار. فمثلاً، يجب أن يكون في باش (أو في bashrc.) مايلي: export PATH=$PATH. إذا لم يكن كذلك، فالمجلد الحالي ليس موجوداً هناك، وإذا كان كذلك، فأعد تعريفه في المسار الأخير.

يجب أن نتذكر أن كثرة نشاط التقاطعات يمكن أن تؤثر على أداء المعالج مقارنة بالعمليات التي يتم تنفيذها. يمكننا – بالمراقبة (atsar -I) – أن نرى العلاقة بين عدد التقاطعات في الثانية، واتخاذ قرارات تتعلق بالأجهزة المسببة لها. على سبيل المثال، تغيير المودم بواحد أذكى، أو تغيير هيكلية الاتصالات إذا اكتشفنا نشاطاً متزايداً على المنافذ التسلسلية الذي تتصل به.

4) استخدام هائل للقرص:

يمكن أن يكون السبب وراء وقت الاستجابة المنخفض – بعد الذاكرة – نظام الأقراص المستخدَم. بداية، علينا أن نتحقق من أن هناك وقتاً للمعالج (كأن يكون idle < 20% مثلاً) وأن رقم in/out كبير (in/out/s > 30 مثلاً) باستخدام atsar -u و atsar -d. يمكن أن يكون الحلّ:

1. في الأنظمة متعددة الأقراص، يكون ذلك بالتخطيط لمكان اختزان الملفات الأكثر استخداماً، وموازنة الزِحام بينها (كأن نضع /home/ في قرص، و/usr/ في آخر) والتأكد من أنه يمكنهما استخدام كل إمكانيات الدخل/الخرج مع الاختزان المؤقت في نفس الوقت (وذلك مثلاً بتخطيط مكان وضعها في منفذ IDE). ومن ثم التأكد من موازنة الحمل. ومن ثم التأكد من أن الزِحام متوازن باستخدام atsar -d (أو iostat). في الأوضاع الحرجة، يمكننا التفكير في شراء نظام أقراص RAID، والتي يمكن أن تقوم بهذه الإعدادات آليًّا.

2. أبق في بالك أنه يمكن تحقيق مستويات أداء أفضل باستخدام قرصين صغيرين بدلاً من قرص واحد كبير يساوي حجمه حجم الاثنين معاً.

3. في الأنظمة التي تحوي قرصاً واحداً فقط، وذلك بشكل عام لتقليل الحجم، يتم عمل 4 أقسام بالطريقة التالية: /, /usr/, swap, /home/؛ ولكن هذا ينشئ وقت استجابة سيّئ جدًّا للدخل/الخرج، وذلك بسبب – على سبيل المثال – وجود مستخدم يقوم بتصريف برنامج موجود في /home/user/ عبر مصرّف موجود في /usr/bin/، حيث سيتحرك رأس القرص عبر القرص بأكمله. في هذه الحالة، من الجيد أن يكون القسمان /usr/ و /home/ موجودين على قرص واحد (كبير)، رغم أن هذا يتسبب ببعض الإزعاج عندما يتعلق الأمر بالصيانة.[2]

4. زِد سرعة فيض الاختزان المؤقت (cache buffer) إلى الدخل/الخرج (انظر مثلاً إلى: /proc/ide/hdX/).

5. إذا كنا نستخدم نظام الملفات ext2، فيمكننا استخدام الأمر: dumpe2fs -h /dev/hdX/ للحصول على معلومات عن القرص، و tune2fs /dev/hdX/ لتغيير بعض المعاملات التي يمكن ضبطها للقرص.

6. ومن البديهي أيضاً أنّ تغييرَ القرص إلى واحد ذي سرعة أكبر (عدد الدورات في الثانية RPM)، سيكون دائماً ذا أثر إيجابي على نظام يحدّه دخل/خرج القرص.

5) تحسين النواحي المتعلقة بـ TCP/IP:

اختبر الشبكة بالأمر atsar (أو أيضاً باستخدام netstat -i أو netstat -s | more)، لتحليل ما إذا كانت هناك أيّة حزم مجزوأة أو أخطاء أو حزم متجاهَلة أو فيض ... إلخ، يمكن أن تؤثر على الاتصال، وبالتالي النظام (مثلاً في NFS, NIS, FTP، أو خادم الوِب). إذا تم اكتشاف أية مشكلات، فيمكننا تحليل الشبكة لأخذ أيٍّ من الإجراءات التالية:

١. تجزئة الشبكة باستخدام عناصر فعالة تتخلص من الحزم التي بها مشكلة أو التي ليست لأجهزة في الجزء المعني.

٢. التخطيط للمكان الذي ستكون به الخوادم لتقليل سير البيانات إليها، وبذلك وقت الوصول.

---

2 ما أظن الكاتب يقصده هنا هو أنه كلما زادت مساحة القرص كلما زاد عدد رؤوس القراءة غالباً، وبالتالي تزيد – نظرياً – إمكانية عمل تقسيمات في القرص لا ترتبط برأس القراءة نفسه، وهذا يقلل الوقت اللازم لتحرك رأس القراءة بين الأماكن المتفرقة للبيانات، أو ما يعرف بـ seek time عند القراءة من قسمين منها معاً.

٣. ضبط متغيرات النواة (/proc/sys/net/) – مثلاً – للحصول على تحسينات في سير البيانات، استخدم:

echo 600 > /proc/sys/net/core/netdev max backlog (وتكون قيمتها المبدئية 300).

6) أنشطة أخرى على متغيرات النواة:

هناك مجموعة أخرى من المعاملات في النواة يمكن ضبطها للحصول على مستويات أداء أفضل، ورغم ذلك، يفترض أن يتم عمل ذلك بحرص – بأخذ النقطة التي ذكرناها بعين الاعتبار –، منتبهين إلى أنه يمكن أن يأتي ذلك بنتائج عكسية أو تعطيل النظام. راجع التوزيعة من أجل المصدر البرمجي، فالملفات vm.txt و fs.txt و kernel.txt و sunrpc.txt الموجودة في Documentation/sysctl:

١. /proc/sys/vm: تحكم بالذاكرة الافتراضية (VM)للنظام. تجعل الذاكرة الافتراضية من الممكن للعمليات التي لا تقوم بالوصول إلى الذاكرة الرئيسية بأن يقبلها النظام ولكن في جهاز الإبدال swap غير المحدودة بالنسبة للمبرمجين فيما يتعلق بحجم برنامجهم (بداهة، يجب أن يكون أصغر من مساحة الإبدال. المعاملات التي يمكن ضبطها يمكن أن تتغير بسهولة عبر gpowertweak.

٢. /proc/sys/fs: يمكن ضبط معامل التفاعل مع نظام ملفات النواة Kernel-FS، كقيمة file-max.

٣. وأيضاً عبر /proc/sys/kernel و /proc/sys/sunrpc.

7) إنشاء النواة التي تتوافق مع متطلباتنا.

إن تحسين النواة يعني اختيار معاملات التصريف بناء على احتياجاتنا. من المفيد أولاً قراءة الملف في /usr/src/linux. سيجعل الضبط الجيد للنواة من الممكن لها أن تعمل أسرع، وإعطاء ذاكرة أكبر لعمليات المستخدم، وجعل النظام ككل أكثر استقراراً. هناك طريقتان لبناء نواة: أحادية (مستويات أداء أفضل) أو مجزّأة (معتمدة على وحدات، وستكون هناك توافقية أعلى إذا كان لدينا نظام هجين بشكل كبير ولم نكن نرغب بتصريف نواة لكل منها). لتصريف نواتك

الخاصة وتطويعها لتناسب احتياجات عتادك، فلكل توزيعة قواعدها الخاصة (رغم أن الإجراءات مشابهة).

8) المقالات التالية مفيدة جدًّا:

لمعلومات عن تحسين وتضبيط أنظمة خوادم لينكس

http://people.redhat.com/alikins/system_tuning.html

أدوات مراقبة أداء للينكس، رغم أنها مقالة قديمة وأن بعض الخيارات غير متاحة، فستبقى المنهجية نفسها قابلة للاستخدام.

http://www.linuxjournal.com/article.php?sid=2396

1.3 تحسينات عامة

هناك مجموعة من التحسينات العامة يمكن أن تحسن أداء النظام:

١) مكتبات ثابتة أو ديناميكية: عندما يتم تصريف برنامج، يمكن عمل ذلك بمكتبة ثابتة (libr.a) أكوادها العاملة مضمنة في الملف التنفيذي، أو مكتبة ديناميكية (libre.so.xx.x)، بحيث يتم تحميلها في وقت التنفيذ. رغم أن الاولى تضمن كوداً محمولاً وآمناً، إلا أنها تستهلك ذاكرة أكبر. يجب أن يقرر المبرمج أي الخيارين مناسب أكثر لبرنامجه، بتضمين الخيار static- في خيارات المصرّف (عدم إضافة هذا الخيار يعني أان المكتبة ستكون ديناميكية)، أو disable-shared-- عند استخدام أمر الضبط. يُنصح باستخدام المكتبة المعيارية libc.a و libc.so (تقريباً كلّ التوزيعات الحديثة تفعل هذا) بإصدار حديث (والتي تعرف بـ libc6) التي تستبدل سابقاتها.

٢) اختيار المعالج المناسب: إنشاء كود تنفيذي للمعمارية التي ستعمل عليها التطبيقات. بعض أكثر معاملات المصرّف تأثيراً: march- (على سبيل المثال، march i686- أو march k6-) وذلك ببساطة بكتابة gcc -march i686، فإن معامل التحسين O1,2,3 (حيث سيُنشئ O3- أسرع إصدار من البرنامج، gcc -O3 -march = i686)، والمعاملات f- (راجع التوثيق للأنواع المختلفة).

٣) تحسين الأقراص: في الوقت الحالي، تتضمن معظم الحواسيب قرص UltraDMA – 100 مبدئياً، ورغم هذا، ففي

معظم الحالات، هي غير محسنة لتقديم أفضل مستوى أداء. هناك أداة (تُدعى hdparm) يمكن استخدامها لتضبيط النواة بمعاملات الأقراص من نوع IDE[3]. علينا أن نكون حذرين عند استخدام هذه الاداة، خاصة في أقراص UltraDMA (تفحص BIOS للتأكد من أن معاملات دعم DMA مفعلة)، حيث يمكن أن تعطّل القرص. تفقد المراجع والتوثيق (و man hdparm) لمعرفة أهم التحسينات (والمخاطر التي تترتب عليها)، فعلى سبيل المثال: c3,- d1, -X34, -X66, -X12, -X86, -mXX, -a16, -u1, -W1, -k1, -K1-. يعني كل خيار نوعاً من التحسينات، ويترتب على بعضها مخاطرة عالية، مما يعني أنه يجب علينا أن نعرف القرص جيداً. لمراجعة المعاملات المحسنة، يمكن أن نستخدم hdparm -vtT /dev/hdx (حيث X هو القرص المحسن)، واستدعاء hdparm مع كل المعاملات التي يمكن استخدامها في etc/init.d/ لتحميلها في الإقلاع.

1.4 إعدادات إضافية

هناك المزيد من الإعدادات التكميلية من وجهة النظر الأمنية التي يمكن تقديمها بالتحسينات، لكنها ضرورية بشكل أكبر عندما يكون النظام متصلاً بشبكة داخلية أو بالإنترنت. تتطلب هذه الإعدادات المهام التالية:

١. تعطيل الإقلاع أو نظام التشغيل الآخر: إذا كان لدى أحد ما وصولاً فيزيائيًّا إلى الجهاز، فسيكون بإمكانه الإقلاع بنظام تشغيل آخر مضبوط مسبقاً وتعديل نظام التشغيل الموجود، مما يعني أنه يجب علينا الوصول إلى إعدادات BIOS لتعطيل الإقلاع باستخدام الأقراص القابلة للإزالة وضبط كلمة مرور (تذكر كلمة مرور BIOS، وإلا فستواجه مشاكل عندما ترغب بتغيير الإعدادات).

٢. الإعداد والشبكة: يُنصح بقطع الاتصال مع الشبكة عندما نضبط النظام. يمكنك إزالة الكابل أو تعطيل الجهاز بالأمر / etc/init.d/networking stop (و start لإعادة تفعيلها) أو بـ ifdown eth0 (استخدم ifup لتفعيلها) لأي جهاز بعينه.

---

3 أقراص IDE هي نوع قديم من الأقراص الصلبة ذات سرعة منخفضة نسبياً، وتأتي وصلة IDE على شكل كابل عريض مكون من مجموعة من الأسلاك وله من الطرفين مقبس بأربعين خطاً، ويمكن للمنفذ الواحد وصل قرصين والتعامل معهما معاً.

٣. عدّل الملفات etc/security/ بما يتناسب مع استخدام النظام والاحتياجات الأمنية. على سبيل المثال، في access.conf لمن يمكنه الولوج إلى النظام

```
Format: permission:          users     : origins +o - : users: from where
-:ALL EXCEPT root: tty1        Disable access to all no-root over tty1.
-:ALL EXCEPT user1 user2 user3:console            prevents access except for
users1,2,3 but the latter may only access from the console.
-:user1:ALL EXCEPT LOCAL .uoc.edu 'group.conf':
```

علينا أيضاً أن نضبط المجموعة للتحكم بماهية وكيفية وأيضاً الحد الأقصى (limits.conf) لإنشاء الحدّ الأقصى لعدد مرات استخدام CPU أو I/O إلخ. لتجنب هجمات DoS مثلاً.

٤. حافظ على أمن كلمات مرور المستخدم الجذر: استخدم 6 محارف على الأقل، بحرف واحد كبير على الأقل، وبعض الرموز الأخرى (.-_,)؛ هذا ليس أمراً سخيفاً، وكذلك، يُنصح بتفعيل خيار انتهاء صلاحية كلمة المرور لإجبار نفسك على تغييرها دورياً، إضافة إلى تحديد عدد المرات التي يمكن لأحد ما أن يدخل فيها كلمات مرور خاطئة. وكذلك، سيكون علينا تغيير المعامل min x في المدخلة في etc/pam.d/passwd/ للإشارة إلى العدد الأدنى للمحارف المستخدمة في كلمة المرور (حيث x هو عدد المحارف).

٥. لا تلج إلى النظام كمستخدم جذر: أنشئ حساباً مثل sysadm واعمل به. إذا كنت تتصل به عن بعد، فسيكون عليك دائماً أن تستخدم ssh للاتصال بـsysadm والقيام بـ – su – إذا كا هذا ضرورياً – للعمل كمستخدم جذر.

٦. اضبط الحد الأقصى لوقت الخمول: فعل المتغير TMOUT – على 360 على سبيل المثال (القيمة المشار إليها بالثواني)، والتي ستكون قيمة أطول مدة لانعدام التفاعل التي ستسمح خلالها الصدفة بالعمل قبل أن تحجبه؛ من الممكن وضعه في ملفات إعداد الصدفة (على سبيل المثال، etc/profile/، و bashrc./~، ...). إذا كنا نستخدم بيئات رسومية، (كدي، جنوم، إلخ)، فعل الخيار لإغلاق حافظة الشاشة بكلمة مرور.

٧. ضبط NFS في الوضع المقيّد: في etc/exports/ صدّر ما هو ضروريّ فقط، دون استخدام الرموز الخاصة wildcards، بالسماح فقط للوصول للقراءة، وعدم السماح بالوصول للكتابة في وضع الجذر، على سبيل المثال،

باستخدام / (ro,root_squash) host.domain.com directory_exported.

٨. تجنب الإقلاع من محمل الإقلاع باستخدام معاملات: يمكن أن يتم إقلاع النظام في وضع single، الذي سيشغل النظام في طور المستخدم الوحيد. اضبط النظام بحيث تكون كلمة السر ضرورية دائماً عند الإقلاع في هذا الوضع. من أجل عمل ذلك، تأكد من أن etc/inittab/ يحوي السطر التالي: S:wait:/sbin/sulogin وأن bin/sulogin/ مفعل. إضافة إلى ذلك، يجب أن يكون لملف إعداد محمل الإقلاع كل الصلاحيات المطلوبة بحيث لا يكون باستطاعة أحد تعديله ما عدا المستخدم الجذر (باستخدام chmod 600). لتجنب أيّ تغييرات بالخطأ، غير خاصية ال blocking باستخدام chattr +i (استخدام i- عندما ترغب بتغيير ذلك). يسمح هذا الملف بمجموعة من الخيارات التي يفترض أن يتم أخذها بعين الاعتبار: timeout، أو – إذا كان في النظام نظام تشغيل واحد فقط للإقلاع مباشرة – restricted، وذلك لمنع الآخرين من التمكن من إدخال أوامر عند الإقلاع مثل linux init = /bin/sh، والحصول على وصول غير مصرح به بالمستخدم الجذر؛ في هذه الحالة، يجب استخدام كلمة المرور؛ إذا أدخلنا فقط كلمة المرور، فسنُسأل عن كلمة المرور لتحميل صورة النواة. هذه الخيارات متاحة في العديد من محملات الإقلاع، مثل lilo و grub-legacy و grub.[4]

٩. التحكم بالمجموعة Ctrl+Alt+Delete. لمنع الآخرين من أن يكونوا قادرين على إطفاء الجهاز من لوحة المفاتيح، أضف رمز التعليق "#" في العمود الأول من السطر التالي:

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

فعل التغييرات بالأمر telinit q

١٠. تجنب الخدمات غير المقدمة: احظر الملف etc/services/ بحيث لا تعترف بخدمات لم يتم التحقق منها بحظر الملف بالأمر chattr +i /etc/services.

١١. اتصال الجذر: عدل الملف etc/securetty/ الذي يحوي TTY و VC (أي Virtual Console) الذي يمكن للجذر الاتصال عبره، بترك واحد من كل منها – مثل tty1 و vc/1 – وإذا كان من الضروري الاتصال عبر sysadm،

---

4 بتصرف

فنفّذ su.

١٢. تخلص من حسابات المستخدمين غير المستخدمة: احذف المستخدمين والمجموعات غير الضروريين، بما فيها ما يأتي مبدئيًّا (مثل operator, shutdown, ftp, uucp, games ...) واترك فقط الضرورية منها (root, bin, daemon, sync, nobody, sysadm) وتلك التي تم إنشاؤها بتثبيت الحزم أو باستخدام أوامر (نفس الأمر مع /etc/group). إذا كان النظم حيوياً، فقد نفكر بحظر (chattr +i) الملفات /etc/passwd و /etc/shadow و /etc/group و /etc/gshadow لتجنب تعديلها (كن حذراً عند القيام بهذه العملية، لأنك بالتالي لن تكون قادراً على تغيير كلمة المرور).

١٣. ضمّ الأقسام بطريقة مقيّدة: استخدم في /etc/fstab طرقاً للأقسام مثل nosuid (الذي يجعل من الممكن مستخدم أو مجموعة القسم)، و nodey (الذي لا يفسر الجهاز المحرفي أو الكلي على ذلك القسم)، و noexec (الذي لا يسمح بتنفيذ الملفات على ذلك القسم). على سبيل المثال:

```
/tmp /tmp ext2 defaults,nosuid,noexec 0 0
```

يُنصَح أيضاً بضم /boot/ على قسم منفصل بمعاملات قراءة فقط.

١٤. حمايات عديدة: غير حمايات الملفات في /etc/init.d/ (خدمات النظام) إلى 700 بحيث يكون بإمكان الجذر وحده تعديلها و تشغيلها و إيقافها، وتعديل الملفات /etc/issue و /etc/issue.net بحيث لا تعطي أيّة معلومات (نظام التشغيل، الإصدار، ...) عندما يتصل أحد ما باستخدام telnet، أو ssh، إلخ.

١٥. SUID و SGID: يمكن للمستخدم أن ينفذ أمراً كمالك إذا كانت الخانة SUID أو SGID لديه مفعلة، مما قد يعكس وجود s لـ SUID (مثل -rwsr-xr-x) و SGID، (مثل -r-xr-sr-x). ولهذا، من الضروري حذف الخانة الثنائية (chmod a-s) من الأوامر التي لا تحتاجها. يمكن البحث عن هذه الملفات بالأمر

```
find / -type f -perm -4000 or -perm -2000 –print
```

يجب أن نستمر بحذر فيما يتعلق بالملفات التي نزيل منها SUID و SGID، حيث يمكن أن يعطّل ذلك الأمر.

١٦.الملفات المشبوهة: يجب أن تتفقد دورياً الملفات ذات الأسماء غير الاعتيادية، والملفات المخفية، والملفات التي ليس لها uid/gid صالح، مثل "..." (ثلاث نقاط)، و ".. " (نقطتان ثم فراغ)، و "..^G" ، ومن أجل هذا، ستحتاج لاستخدام:

```
find / -name ".*" -print | cat -v
```

وإن لم يكن، فبهذا:

```
find / name ".." -print
```

ولاكتشاف uid/gid غير صالحة، استخدم find / -nouser أو nogroup- (كن حذراً، لأن بعض التثبيتات يتم تنفيذها بمستخدم لا يتم التعرف عليه فيما بعد، وعلى المدير أن يتغير).

١٧.الاتصال دون كلمة مرور: لا تسمح بالملف rhosts. في أيّ مستخدم مالم يكن هذا ضرورياً جدًّا (ننصح باستخدام ssh بكلمة مرور عامة بدلاً من هذه الطريقة المعتمدة على rhosts).

١٨.مدير الواجهة الرسومية X Display manager: عدل الملف etc/X11/xdm/xaccess/ لتحديد المضيفات التي يمكنها الاتصال عبر XDM وتجنب أيّ مضيفات لها شاشة ولوج[5].

1.5 المراقبة

هناك أداتان مفيدتان جدًّا لمراقبة النظام: وهما monin و monit. يُنتج munin رسماً يوضع المعاملات المختلفة للخادم (معدل الحمل، استخدام الذاكرة، استخدام المعالج، تمرير محتوى MySQL، وسير بيانات eth0، إلخ) دون إعدادات كثيرة، حيث يتحقق monit من توفر الخدمات، مثل Apache و MySQL و Postfix، ويتضمن أنشطة عديدة، كإعادة تفعيل خدمة غير متوفرة. توفر التجميعة رسومات هامة للتعرف على مكان نشأة المشكلات وما يُنشئها.

5 إن XDM هو أحد مدراء الولوج، وهناك غيره، مثل GDM و KDM و LightDM وغيرها. للمزيد عن هذا الموضوع، راجع توثيق مدير الولوج الذي لديك.

لنفترض أن نظامنا يُدعى pirulo.org، وأن صفحتنا مضبوطة في www.pirulo.org، والمستندات في /var/www/pirulo.org/web. لتثبيت munin على ديبان، يمكننا تنفيذ – على سبيل المثال – apt-get install munin munin-node.

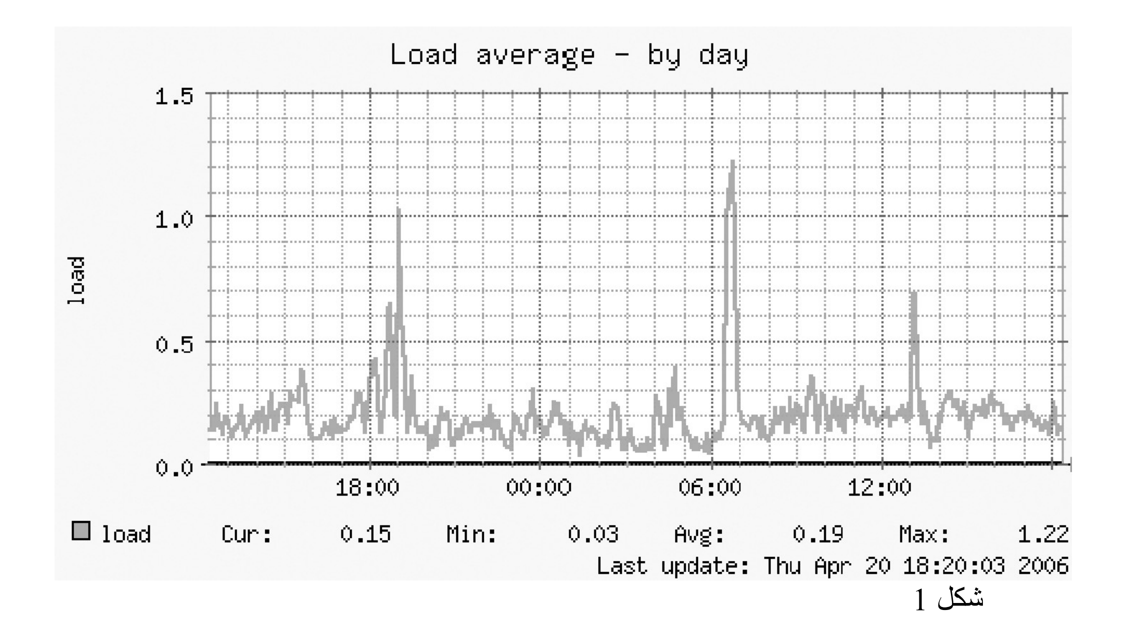
يجب علينا بعد ذلك ضبط munin (الملف /etc/munin/munin.conf) كالتالي:

```
dbdir /var/lib/munin
htmldir /var/www/www.pirulo.org/web/monitoring
logdir /var/log/munin
rundir /var/run/munin
tmpldir /etc/munin/templates
[pirulo.org]
address 127.0.0.1
use_node_name yes
```

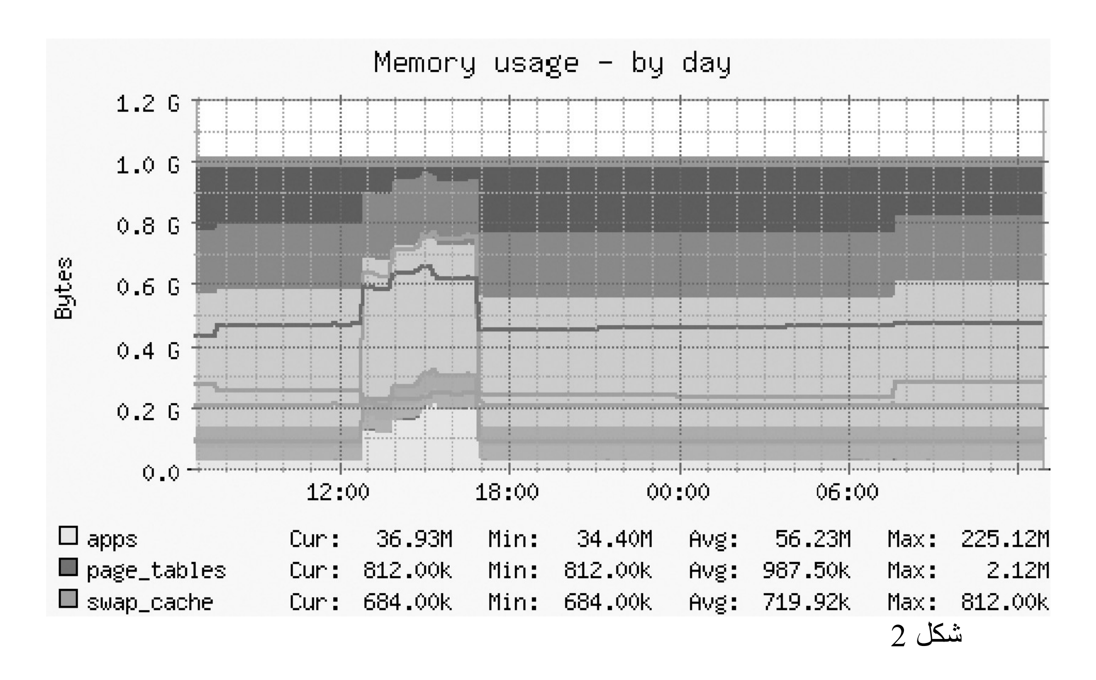
ثم يتم إنشاء المجلد، وتغيير الصلاحيات، وتشغيل الخدمة.

```
mkdir -p /var/www/pirulo.org/web/monitoring
chown munin:munin /var/www/pirulo.org/web/monitoring
/etc/init.d/munin-node restart
```

سنكون بعد بضع دقائق قادرين على رؤية المخرجات الأولى في http://www.pirulo.org/monitoring في المتصفح. ونعرض أدناه رسمين (الحمل والذاكرة) كمثال.



شكل 1



شكل 2

إذا كنت ترغب بالحفاظ عل الخصوصية في الرسومات، فكل ما عليك فعله هو ضبط كلمة سر للوصول إلى المجلد عبر أباتشي. على سبيل المثال، يمكننا حفظ الملف htaccess. بالمحتويات التالية في المجلد :/var/www.pirulo.org/web/monitoring

```
AuthType Basic
AuthName "Members Only"
AuthUserFile /var/www/pirulo.org/.htpasswd
<limit GET PUT POST>
require valid-user
</limit>
```

علينا بعد ذلك أن ننشئ ملف كلمة السر في /var/www/pirulo.org/.htpasswd بالأمر (كمستخدم جذر):

```
htpasswd -c /var/www/pirulo.org/.htpasswd admin
```

عندما نتصل بـ www.pirulo.org/monitoring، فلن يسأل عن اسم المستخدم (admin) وكلمة السر التي أدخلناها بعد الأمر السابق.

لتثبيت monit، ننفذ apt-get install monit، و نعدل /etc/monit/monitrc. يتضمن الملف المبدئي مجموعة من الأمثلة، ولكن يمكننا الحصول على المزيد من http://www.tildeslash.com/monit/doc/examples.php. على سبيل المثال، إذا رغبنا بمراقبة proftpd, sshd, mysql, apache, postfix، فيمكننا كتابة التالي في monitrc لتفعيل واجهة الوب

لـ monit على المنفذ 3333:

```
set daemon 60
set logfile syslog facility log_daemon
set mailserver localhost
set mail-format { from: monit@pirulo.org }
set alert root@localhost
set httpd port 3333 and
allow admin:test

        check process proftpd with pidfile /var/run/proftpd.pid
        start program = "/etc/init.d/proftpd start"
        stop program = "/etc/init.d/proftpd stop"
        if failed port 21 protocol ftp then restart
        if 5 restarts within 5 cycles then timeout

        check process sshd with pidfile /var/run/sshd.pid
        start program "/etc/init.d/ssh start"
        stop program "/etc/init.d/ssh stop"
        if failed port 22 protocol ssh then restart
        if 5 restarts within 5 cycles then timeout

        check process mysql with pidfile /var/run/mysqld/mysqld.pid
        group  database
        start program = "/etc/init.d/mysql start"
        stop program = "/etc/init.d/mysql stop"
        if failed host 127.0.0.1 port 3306 then restart
        if 5 restarts within 5 cycles then timeout

        check process apache with pidfile /var/run/apache2.pid
        group  www
        start program = "/etc/init.d/apache2 start"
        stop program = "/etc/init.d/apache2 stop"
        if failed host www.pirulo.org port 80 protocol http and request "/monit/token" then
        restart
        if cpu is greater than 60% for 2 cycles then alert
        if cpu > 80% for 5 cycles then restart
        if totalmem > 500 MB for 5 cycles then restart
        if children > 250 then restart
        if loadavg(5min) greater than 10 for 8 cycles then stop
        if 3 restarts within 5 cycles then timeout
```

```
check process postfix with pidfile /var/spool/postfix/pid/master.pid
group  mail
start program = "/etc/init.d/postfix start"
stop program = "/etc/init.d/postfix stop"
if failed port 25 protocol smtp then restart
if 5 restarts within 5 cycles then timeout
```

راجع أدلة الاستخدام لمزيد من المعلومات http://www.tildeslash.com/monit/doc/manual.php. للتحقق من أن خادم أباتشي تعمل مع monit، علينا أن نضع الإعداد الذي يصل إلى if failed host www.pirulo.org port 80 protocol http and request “/monit/token” then restart. إذا لم يكن بإمكاننا الوصول إليه، فهذا يعني أن أباتشي لا يعمل، مما يعني أنه يجب أن يوجد هذا الملف (mkdir /var/www/pirulo.org/web/monit; echo "pirulo" > /var/www/pirulo.org/web/monit/token). من الممكن أيضاً ضبط monit بحيث يعمل مع SSL (انظر إلى http://www.howtoforge.com/server_monitoring_monit_munin_p2).

وفي النهاية، يجب علينا أن نعدّل /etc/default/monit لتفعيل monit وتغيير startup=1 و CHECK_INTERVALS=60 على سبيل المثال (بالثواني). إذا قمنا بتشغيل monit (عبر /etc/init.d/monit start) واتصلنا بـ http://www.pirulo.org:3333، فسنرى شاشة تشبه ما يلي:

Monit Service Manager

Process	Status	Uptime	CPU	Memory
proftpd	runnig	1h 19m	0,0%	1,2% [2348 Kb]
sshd	runnig	1h 56m	0,0%	0,7% [1508 Kb]
mysql	runnig	1h 29m	0,0%	7,1% [13664Kb]
apache	runnig	15m	0,0%	5,0% [9628 Kb]
postfix	runnig	1h 26m	0,0%	0,6% [1322 Kb]

Process status

Parameter	Value
Name	apache
Pid file	/var/runapache2.pid
Status	running
Group	www
Monitoring mode	active
Monitoring status	monitored
Start program	/etc/nt.d/apache2 start
Stop program	/etc/nt.d/apache2 stop

شكل 3

هناك أدوات أكثر تعقيداً لمراقبة الشبكة وخدماتها باستخدام ميفاق إدارة الشبكات البسيط SNMP و multi-route traffic grapher (أو MRTG) مثلاً. يمكن العثور على مزيد من المعلومات حول هذا الموضوع في <http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance>.

تم إنشاء MRTG (في http://oss.oetiker.ch/mrtg) بشكل أساسي لجعل معلومات الشبكة مرئية، ولكن يمكن استخدام بيانات أخرى لجعل سلوكها مرئياً، لإنشاء إحصائيات معدل الحمل في الخادم مثلاً. ولعمل هذا، نستخدم الحزم mrtg و atsar. بمجرد تثبيتها، سنضبط الملف /etc/mrtg.cfg:

```
WorkDir: /var/www/mrtg

Target[average]: '/usr/local/bin/cpu-load/average'

MaxBytes[average]: 1000

Options[average]: gauge, nopercent, growright, integer

YLegend[average]: Load average

kMG[average]: „

ShortLegend[average]:

Legend1[average]: Load average x 100

LegendI[average]: load:

LegendO[average]:

Title[average]: Load average x 100 for pirulo.org

PageTop[average]: <H1>Load average x 100 for pirulo.org</H1>

<TABLE>

<TR><TD>System:</TD> <TD>pirulo.org</TD></TR>

<TR><TD>Maintainer:</TD> <TD>webmaster@pirulo.org</TD></TR>

<TR><TD>Max used:</TD> <TD>1000</TD></TR>

</TABLE>
```

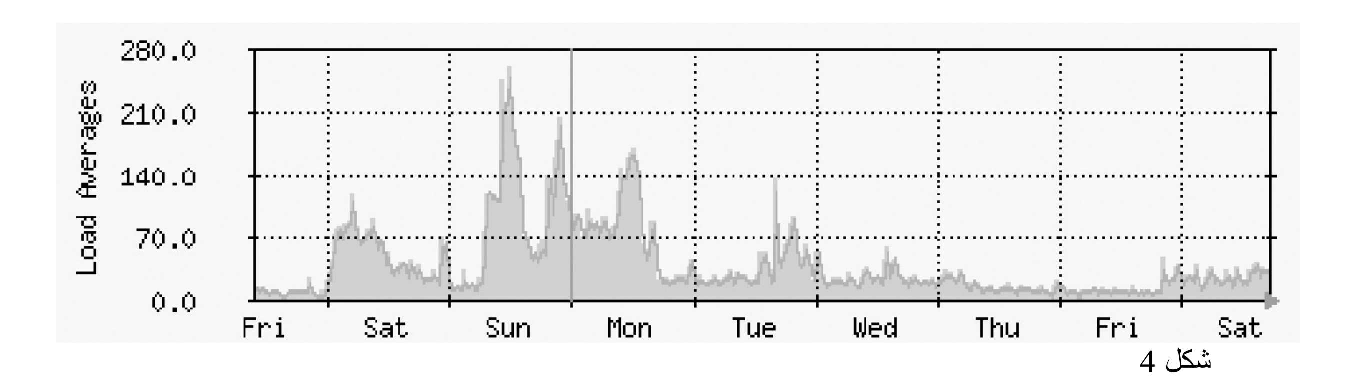
لإنشاء البيانات في atsar (أو sar)، ننشئ نصاً برمجياً في /usr/local/bin/cpu-load/average (الذي يجب أن يكون عليه صلاحيات تنفيذ للجميع) الذي سيقوم بتمرير البيانات إلى mrtg:

```
#!/bin/sh

load='/usr/bin/atsar -u 1 | tail -n 1 | awk -F" " '{print $10}'

echo "$load * 100" | bc | awk -F"." '{print $1}'
```

يجب أن ننشئ ونغير التصاريح في المجلد var/www/mrtg/. يتم تنفيذ mrtg مبدئياً في cron، لكن إذا رغبنا في تنفيذه، فيمكننا تشغيل etc/mrtg.cfg/، وسينشئ هذا الرسومات في var/www/mrtg/average.html/ والذي يمكن رؤيته عبر المتصفح على الرابط http://www.pirulo.org/mrtg/averange.html.



شكل 4

ومن الحزم الأخرى المثيرة للاهتمام والتي يفترض أخذها بعين الاعتبار عند مراقبة النظام ما يلي:

- Frysk (الموقع http://sources.redhat.com/frysk): إن الهدف من مشروع frysk هو إنشاء نظام مراقبة قابل للتوزيع وذكي لمراقبة العمليات وخيوط المعالجة.

- Cacti (الموقع http://cacti.net). إن Cacti حل رسومي مصمم للعمل مع بيانات PRDTools. يقدم Cacti أشكالاً مختلفة من الرسومات ووسائل الحصول على المعلومات و الخصائص التي يمكن للمستخدم التحكم بها بسهولة شديدة، وهو نظام مطوع للعمل في جهاز أو حتى في بيئة معقدة من الأجهزة والشبكات والخوادم.

سنعرض الآن أدوات أخرى لا تقل أهمية عنها (بترتيب أبجدي) يتضمنها جنو/لينكس (كديبان مثلاً) لمراقبة النظام. هذه ليست قائمة طويلة، ولكنها ببساطة مجموعة منتقاة من أكثر الأدوات المستخدمة شيوعاً (ننصح بالاطلاع على دليل استخدام كل منها لمزيد من المعلومات):

- atsar, ac, sac, sysstat, isag[6] : أدوات تدقيق لتحليل الموارد العتادية والبرمجية.

- Arpwatch و mon: مراقب نشاط Ethernet/FDDI يشير إلى التغييرات في جداول MACIP في حال حدوثها؛ مراقب خدمات شبكة.

- Diffmon, fcheck: يُنشئ تقريراً بالتغييرات على إعدادات النظام ويراقب نظام الملفات بحيث يكتشف الاختراق حال حدوثه.

- Fam: اختصار لـ File Alteration Monitor، أي مراقب تبديل الملفات.

- Genpower: مراقب لإدارة المشاكل في مصدر الطاقة.

- Gkrellm: مراقب رسومي للمعالج، والعمليات (الذاكرة)، وأنظمة الملفات، والمستخدمين، والقرص، والشبكات والإنترنت، والإبدال swap، إلخ.

- Ksensors: (أو lm-sensors): مراقب للوحة الأم (درجة الحرارة، ومصدر الطاقة، والمراوح، إلخ).

- Systune, .lcap: يتخلص من إمكانيات مفعلة في الملف proc/sys/kernel/ ويطوعها لتناسب احتياجات المستخدم.

- Log watcher: محلل تقارير.

- Munin و monit: مراقبة رسومية للنظام.

- Powertweak و gpowertweak: يراقب ويعدل المعاملات المختلفة للعتاد، والنواة، والشبكة، و VFS، و VM (يسمح بتعديل بعض الخيارات المعروضة سابقاً في /proc/).

- Gps, gtop, tkps, lavaps: (مرتبة – من اليسار إلى اليمين – من الأكثر إلى الأقل سلاسة في الاستخدام): أنواع عديدة من مراقبات العمليات (وبشكل عام تستخدم معلومات من /proc/) وتسمح لنا برؤية الموارد، والمقابس، والملفات،

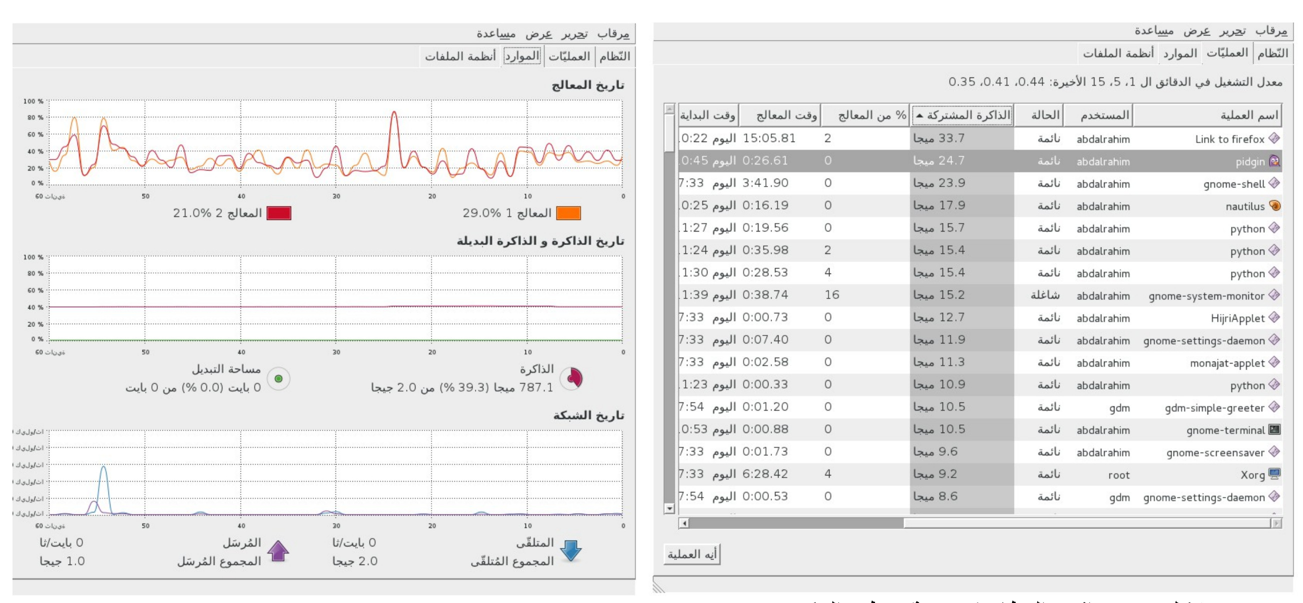
---

6 إن isag اختصار لـ Interactive System Activity Grapher.

والبيئة والمعلومات الأخرى التي تستخدمها، إضافة إلى إدارة مواردها وحالاتها.

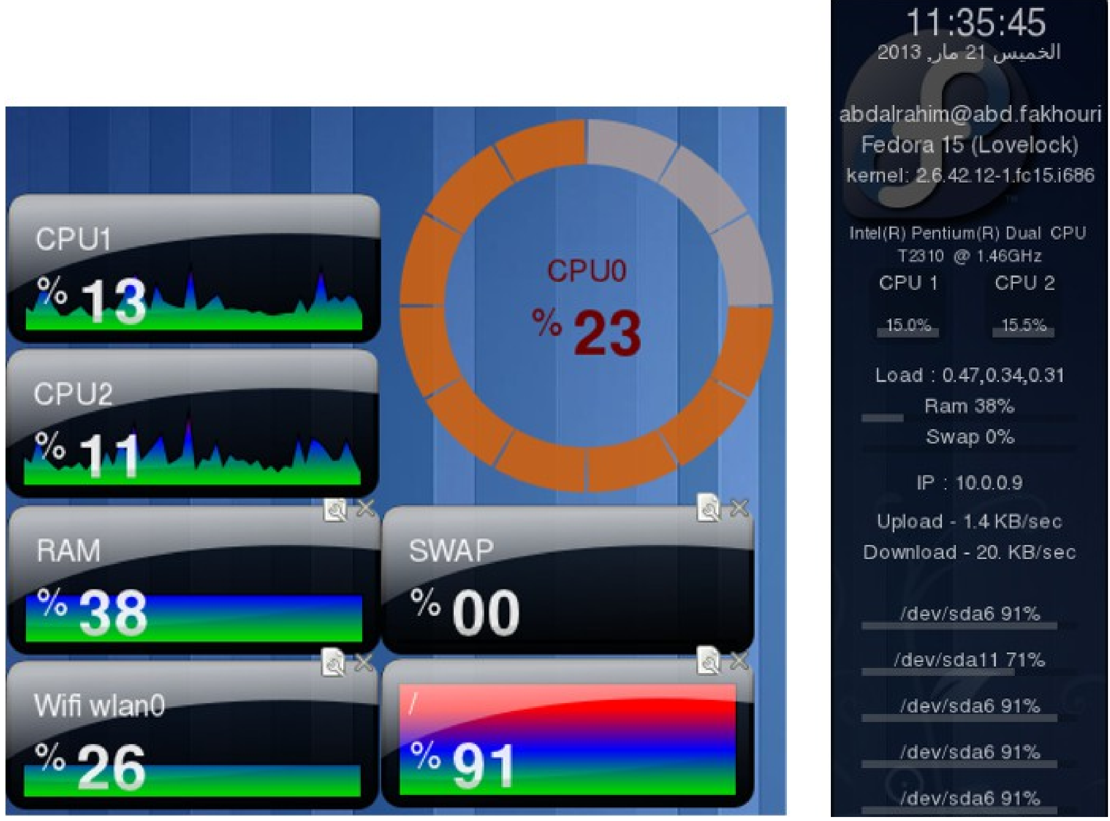
- Swatch: مراقبة أنشطة النظام عبر ملفات التقارير.
- Vtgrab: مراقبة الأجهزة البعيدة (مشابهة لـ VNC).
- Whowatch: أداة لمراقبة المستخدمين في الوقت الحقيقي.
- Wmnd, dmachinemon: مراقب لسير البيانات على الشبكة، ومراقبة تجمعات (عناقيد) الأجهزة على الشبكة.
- Xosview, si: مراقب رسومي للموارد، وعارض لمعلومات النظام.

تعرض واجهات ksensors و gkrellm و xosview النتائج من عملية المراقبة في الوقت الحقيقي. ويظهر فيما يلي تطبيق مراقبة النظام من بيئة سطح المكتب جنوم (وتوفر بيئة كدي كذلك واحداً مثله تقريباً)، والذي يوفر عرضاً بالوقت الحقيقيّ لتغيرات الأداء والعمليات الجارية مع إمكانية فرز العمليات وإدارتها:



شكل 5: مراقب النظام في بيئة سطح المكتب جنوم

وهناك مراقبات مشابهة في معظم بيئات سطح المكتب المتكاملة كذلك. وهناك أيضاً ودجات لمراقبة الأداء، مثل بعض ودجات بيئة كدي وبعض ودجات screenlets المبنية بمكتبات GTK. وفيما يلي صورة لودجات مراقبة النظام من Screenlets:



شكل 6: ودجات مراقبة النظام من screenlets

الأنشطة

١) قم بعملية مراقبة شاملة للنظام باستخدام الأدوات التي تراها أكثر كفاءة في تشخيص استخدام الموارد وظاهرة عنق الزجاجة التي يمكن أن تكون موجودة في النظام. حاكِ حمل النظام بالكود sumdis.c المُعطى في الوحدة التي تغطي الشبكات العنقودية. فعلى سبيل المثال، استخدم:

```
sumdis 1 2000000
```

٢) غير معاملات النواة والمصرّف لتنفيذ الكود المذكور في النقطة السابقة (sumdis.c) باستخدام ما يلي مثلاً:

```
time ./sumdis 1 1000000
```

٣) نفس الأمر مع كلتي النواتين، وقم بالاستنتاج بناء على النتائج.

المراجع

مصادر أخرى للمراجع والمعلومات:

[Debc, Ibi]

تحسين خوادم لينكس: http://people.redhat.com/alikins/system_tuning.html

المراقبة باستخدام Munin و Monit على:

http://www.howtoforge.com/server_monitoring_monit_munin

المراقبة باستخدام SNMP و MRTG على:

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance

Munin: http://munin.projects.linpro.no/

Monit: http://www.tildeslash.com/monit/

MRTG: http://oss.oetiker.ch/mrtg/

Frysk: http://sources.redhat.com/frysk/

Cacti: http://cacti.net/

مراجع عربية:

كتاب "إدارة العمليات والنظام" لصبري عبدالله.

الشـــبكات العــنقودية

د. رِمو سُــپي بُلدريتو

مقدمة

تشير الشبكة العنقودية cluster إلى مجموعة حواسيب تعمل معاً ضمن هدف مشترك. تتكون هذه الحواسيب من عتاد وشبكات اتصال وبرمجيات لتعمل معاً، كما لو كانت جزءاً من نظام واحد. هناك أسباب عديدة تجعلنا بحاجة لضبط هذه الشبكات العناقيد، وأهمها التمكن من معالجة البيانات بكفاءة وسرعة أكبر، كما لو كانت نظاماً واحداً. وبشكل عام، يعمل العنقود على شبكة محلية LAN، ويقدم اتصالاً كفؤاً، وتكون الأجهزة فيه قريبة من بعضها فيزيائياً. وهناك مفهوم أشمل وهو الحوسبة الموزعة grid، والذي يتشابه فيه الهدف مع العناقيد، ولكنها تتصل عبر شبكات واسعة النطاق WAN. يعتبر بعض المبرمجين الحوسبة الموزعة كعنقود العناقيد بمفهوم "عالمي". رغم أن التقنيات التي تتقدم بشكل متسارع والتكاليف التي تنخفض باستمرار تجعل ضبط هذه الأنظمة أسهل، إلا أن التعقيد والجهود المطلوبة لاستخدام عشرات أو مئات (وفي بعض الحالات آلاف) الحواسيب كبيران جداً. ورغم هذا الوضع، فإن الفوائد المتعلقة بوقت الحوسبة تجعل هذا النوع من حلول الحوسبة عالية الأداء High Performance Computing – HPC جذاباً وسريع التطور. سنعرض في هذه الوحدة بعضاً من أكثر التوجهات انتشاراً واستخداماً في هذا المجال.

1 مدخل إلى الحوسبة عالية الأداء HPC

لقد نتج عن التطور التقني معالجات وشبكات سريعة وقليلة التكلفة وذات كفاءة عالية، مما نجم عنه توجه نحو التغيير في نسبة الكلفة إلى الأداء لصالح أنظمة المعالجة المتصلة ببعضها أمام استخدام معالج وحيد عالي السرعة. يمكن تصنيف هذا النوع من المعماريات ضمن إعدادين أساسيين:

- أزواج الأنظمة المتلاصقة: وهذه الأنظمة التي يرى فيها المبرمجُ الذاكرةَ مشتركة بين كل المعالجات (أنظمة الذاكرة المشتركة) وذاكرة كل معالج كذاكرة واحدة.
- أزواج الأنظمة المتباعدة: لا تتشارك الذاكرة (لكل معالج ذاكرته) وتتواصل عبر رسائل يتم تمريرها في شبكة (أنظمة تمرير الرسائل).

يعرف النوع الأول بنظام المعالجة المتوازية، ويعرف الثاني بأنظمة الحوسبة الموزعة. في الحالة الثانية، يمكننا القول بأن النظام الموزع هو مجموعة من المعالجات المتصلة فيما بينها على شبكة كلّ معالج فيها له موارده الخاصة (الذاكرة والطرفيات) وتتواصل بتبادل الرسائل على الشبكة.

إن أنظمة الحوسبة ظاهرة حديثة نسبياً (يمكننا القول بأن تاريخ الحوسبة قد بدأ في السبعينات). لقد كانت في البداية مكونة من أنظمة ضخمة وثقيلة ومكلفة، ويمكن فقط لبعض المختصين استخدامها، وقد كانت بطيئة ويصعب الوصول إليها. وفي السبعينات، أدت التطورات التقنية إلى بعض التحسينات التكوينية التي تم القيام بها باستخدام الوظائف التفاعلية، ومشاركة الوقت، والطرفيات، وقد تم تصغير الحواسيب بشكل كبير. وقد عرفت الثمانينات من القرن الماضي تحسينات في الأداء والكفاءة (والتي استمرت حتى الآن)، وتقليل مستمر للحجم، وإنشاء الحواسيب المصغّرة. لقد استمر تطور الحواسيب ليشمل محطات العمل والتحسينات في الشبكة (من الشبكات المحلية ذات 10ميجابت/ثانية ومودمات 56 كيلوبت/ثانية في 1973، إلى الشبكات المحلية ذات 1 جيجا ووضع النقل غير المتزامن (ATM) في الشبكات واسعة النطاق ذي 1,2جيجابت/ثانية هذه الأيام)، وهذا نظام أساسي في تطبيقات الوسائط المتعددة الحالية والتي سيتم تطويرها في المستقبل القريب. وقد نشأت الأنظمة الموزعة في السبعينات (أنظمة بأربعة أو ثمانية حواسيب)، ولكنها انتشرت بشكل عملي في التسعينات.

رغم أن إدارة وتثبيت وصيانة الأنظمة الموزعة عملية معقدة – علماً بأنها تكبر باستمرار –، فإن الأسباب الأساسية لشعبيتها هي التحسن في الأداء والكفاءة التي تقدمها التطبيقات الموزعة الأصيلة (بسبب طبيعتها)، والمعلومات التي يمكن أن تتشاركها مجموعة من المستخدمين، ومشاركة الموارد، والتسامح العالي مع الأخطاء، وإمكانية التوسعة المستمرة (إمكانية إضافة مزيد من العقد لتحسين الأداء والكفاءة تدريجياً وباستمرار).

سنلقي في الأقسام التالية نظرة على بعض أكثر أنظمة المعالجة المتوازية/الموزعة شيوعاً، إضافة إلى نماذج البرمجة المستخدمة لإنشاء أكواد يمكنها استخدام هذه المزايا.

1.1 بيولف – Beowulf

إن بيولف معمارية متعددة الحواسيب يمكن استخدامها في التطبيقات المتوازية/الموزعة (APD). يتكون النظام بشكل أساسي من خادم وعميل واحد أو أكثر متصلين (بشكل عام) عبر شبكة إثرنت، دون استخدام أي عتاد خاص. لاستكشاف إمكانية المعالجة هذه، فمن الضروري للمبرمجين أن يكون لديهم نموذج برمجة موزعة يمكن أن يتطلب لغة سي واستدعاءات نظام – رغم أنه من الممكن عمل ذلك عبر يونكس (المقابس sockets و RPC) – على سبيل المثال؛ لكن يمكن اعتبار هذه الطريقة منخفضة المستوى.

إن طبقات البرمجيات التي توفرها الأنظمة مثل الأجهزة الافتراضية المتوازية – Parallel Virtual Machines PVM، وواجهة تمرير الرسائل Message Passing Interface – MPI تسهل بشكل كبير استخراج النظام وجعل برمجة تطبيقات متوازية/موزعة بسهولة وبساطة أمراً ممكناً. إن نموذج العمل الأساسي هو معلّم-عمّال (master-workers)، والذي فيه خادم يوزع المهام التي يؤديها العمّال. في الأنظمة الكبيرة (الأنظمة ذات 1024 عقدة)، هناك أكثر من معلم، وعقد مخصصة لمهام محددة، كمراقبة الدخل/الخرج مثلاً.

من الاختلافات الأساسية بين بيولف وعنقود محطات العمل Cluster of Workstations – COW، هو أن بيولف يُرى كجهاز واحد يتم فيه الوصول إلى العقد عن بعد، حيث لا يكون لها طرفية (أو محطة عمل)، بينما COW مجموعة من الحواسيب التي يمكن أن يستخدمها كل من مستخدمي COW وغيرهم تفعلياً عبر شاشة ولوحة مفاتيح. علينا أن نبقي في بالنا أن

بيولف ليس برمجية تحول أكواد المستخدم إلى أخرى موزعة أو تؤثر على نواة نظام التشغيل (مثل Mosix على سبيل المثال). إنه ببساطة طريقة لعمل عنقود من الأجهزة التي تشغّل جنو/لينكس وتتصرف كحاسوب خارق. ومن البديهي أن هناك أدوات تجعل من الممكن الحصول على مكتبة، أو تعديل على النواة، أو ضبط أسهل للحصول على مستويات أداء أفضل، ولكن يمكن أيضاً بناء عنقود بيولف من معايير جنو/لينكس والبرمجيات التقليدية. إن إنشاء عنقود بيولف بعقدتين – مثلاً – يمكن أن يتحقق ببساطة بوجود جهازين متصلين عبر شبكة محلية باستخدام موزع (hub) وتوزيعة جنو/لينكس (دبيان [مثلاً])، ونظام الملفات الشبكي NFS، وبعد تفعيل خدمات الشبكة مثل rsh أو ssh. في وضع كهذا، يمكننا أن نقول بأنه لدينا عنقود بسيط من جهازين.

1.1.1 كيف نضبط العُقَد؟

بداية، يجب علينا أن نضبط etc/hosts/ (في كل عقدة) بحيث يحوي سطر localhost العنوان 127,0,0,1 فقط، ولا يشمل أي اسم للجهاز، كما يلي:

```
127،0،0،1      localhost
```

ثم أضف عناوين العُقد (لكل العقد)، على سبيل المثال:

```
192،168،0،1    pirulo1
192،168،0،2    pirulo2
…
```

من الممكن إنشاء مستخدم (nteum) في كل العقد، وإنشاء مجموعة ثم إضافته إليها:

```
groupadd beowulf
adduser nteum beowulf
echo unmask 007 >> /home/nteum/.bash_profile
```

وبهذه الطريقة يصير من الممكن أن يعدل عنقود beowulf أي ملف ينشئه المستخدم nteum أو أي أحد ضمن المجموعة beowulf.

علينا إنشاء خادم NFS (وستكون بقية العقد عملاء له). على الخادم، ننشئ مجلداً كما يلي:

```
mkdir /mnt/nteum
chmod 770 /mnt/nteum
chown -R nteum:beowulf /mnt/nteum
```

يمكننا الآن تصدير هذا المجلد من الخادم.

```
cd /etc
cat >> exports
/mnt/wolf 192.168.0.100/192.168.0.255 (rw)
<control d>
```

علينا أن نتذكر أن شبكتنا ستكون xxx ,192,168,0 وأنها شبكة خاصة، مما يعني بأن العنقود لن يكون مرئياً من الإنترنت، ويجب علينا ضبط الإعدادات بحيث يمكن للعقد أن ترى بعضها (من جدران الحماية).

علينا التحقق من أن الخدمات تعمل:

```
chkconfig -add sshd
chkconfig -add nfs
chkconfig -add rexec
chkconfig -add rlogin
chkconfig -level 3 rsh on
chkconfig -level 3 nfs on
chkconfig -level 3 rexec on
chkconfig -level 3 rlogin on
```

للعمل بأمان، من المهم العمل مع ssh بدلاً من rsh، مما يعني أنه علينا إنشاء المفاتيح لتوصيل الأجهزة والمستخدم nteum بأمان دون كلمة مرور. لعمل ذلك، نعدّل (بإزالة إشارة التعليق #) عن السطور التالية في /etc/ssh/sshd_config:

```
RSAAuthentication     yes
AuthorizedKeysFile    .ssh/authorized_keys
```

نعيد تشغيل الجهاز ونتصل بالمستخدم nteum، علماً بأن هذا المستخدم سيشغل العنقود. لإنشاء مفاتيح:

```
ssh-keygen -b 1024 -f ~/.ssh/id_rsa -t rsa -N ""
```

سيتم إنشاء الملفين id_rsa و id_rsa.pub في مجلد المكتبات /home/nteum/.ssh، ويجب علينا نسخ الملف id_rsa.pub إلى ملف يُدعى authorized_keys في نفس المجلد. ونعدّل الصلاحيات بالأمرين:

chmod 644 ~/.ssh/aut* و chmod 755 ~/.ssh

علماً بأن العقدة الأساسية فقط سيتم توصيلها بالبقية (وليس العكس)، فسنحتاج فقط لنسخ المفتاح العام (id_rsa.pub) إلى كل عقدة في المجلد/الملف /home/nteum/.ssh/authorized_keys لكل عقدة. إضافة إلى ذلك، سيكون علينا أن نضمّ NFS في كل عقدة بإضافة السطر التالي إلى /etc/fstab:

```
pirulo1:/mnt/nteum /mnt/nteum nfs rw,hard,intr 0 0
```

لقد صار لدينا الآن عنقود Beowulf لتنفيذ تطبيقات يمكن أن تكون PVM أو MPI (سنرى ذلك في القسم التالي). يوجد في فيدورا تطبيق (system-config-cluster) يجعل من الممكن ضبط عنقود عبر واجهة رسومية. لمزيد من المعلومات، ألقِ نظرة على:

http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.html

1.1.2 فوائد الحوسبة الموزعة

ما فوائد الحوسبة المتوازية؟ سنرى ذلك في الأمثلة. لدينا برنامج لجمع أرقام (مثلاً: 4+5+6+ ...) يُدعى sumdis.c مكتوب بلغة سي:

```
#include <stdio.h>
	int main (int argc, char** argv){
float initial, final, result, tmp;

if (argc < 2) {
	printf ("Use: %s N.º initial N.º final\n",argv[0]);
	exit(1);
	}
else {
	initial = atol (argv[1]);
	final = atol (argv[2]);
	result = 0.0;
	}
	for (tmp = inicial; tmp <= final; tmp++){
	result + = tmp; }
printf("%f\n", result)
return 0;
}
```

نصرّفه باستخدام gcc -o sumdis sumdis.c، وإذا نظرنا إلى تنفيذ هذا البرنامج، على سبيل المثال:

time ./sumdis 1 1000000 (من 1 إلى 106)

فسنرى أن الوقت في جهاز يعمل بدبيان 2,4,18 على معالج AMD Athelon 1,400 بذاكرة رام قدرها 256 ميجا يعادل (تقريباً) real = 0,013 و user = 0,010، وبعبارة أخرى، 13 ميلي ثانية ككل، و 10 ميلي ثانية في مساحة المستخدم. بينما إذا أدخلنا

time ./sumdis 1 16000000 (من 1 إلى 106*16)

سيكون الوقت real = 182، وبعبارة أخرى سيكون أكثر بـ 14 ضعفاً، مما يعني أننا إذا نظرنا إلى 160,000,000 (أي $160*10^6$)، فسيكون الوقت من عشرات الدقائق.

إن الفكرة من وراء توزيع الحوسبة هي: إذا كان لدينا عنقود من 4 أجهزة (node1 – node4) بخادم، حيث يتم مشاركة الملفات عبر NFS، فسيكون توزيع الحمل عبر rsh مفيداً (لا ينصح بذلك، لكنه مقبول في مثالنا هذا)، بحيث يجمع الأول من 1 إلى 40,000,000، والثاني من 40,000,001 إلى 80,000,000، والثالث من 80,000,001 إلى

120,000,000,000، والرابع من 120,000,001 إلى 160,000,000. تعرض الأوامر الحالية إحدى الإمكانيات. نعتبر أن للنظام المجلد /home/ المشترك عبر NFS وأن المستخدم (nteum) الذي سينفذ النص البرمجي قد ضبط rhosts. بشكل مناسب بحيث يصير من الممكن الوصول إلى الحساب دون كلمة مرور. إضافة إلى ذلك، إذا تم تفعيل tcpd في /etc/inetd.conf في سطر rsh، فيجب أن يكون الملف ذي العلاقة في /etc/hosts.allow موجوداً، مما يسمح لنا بالوصول إلى الأجهزة الأربعة في العنقود:

```
mkfifo out
```

(يُنشئ طابور fifo في /home/nteum/)

```
./distr.sh & time cat salida | awk '{total + = $1 } END printf "%lf", total}'
```

(ينفذ الأمر distr.sh؛ يتم تجميع وإضافة النتائج أثناء قياس وقت التنفيذ)

يمكن أن يكون النصّ البرمجي distr.sh مشابهاً لما يلي:

```
rsh node1 /home/nteum/sumdis 1 40000000 > /home/nteum/out < /dev/null &
rsh node2 /home/nteum/sumdis 40000001 80000000 > /home/nteum/out < /dev/null &
rsh node3 /home/nteum/sumdis 80000001 120000000 > /home/nteum/out < /dev/null &
rsh node4 /home/nteum/sumdis 120000001 160000000 > /home/nteum/out < /dev/null &
```

يمكننا أن نلاحظ أن الوقت قد انخفض بشكل كبير (بمعامل يقارب 4)، ليس بقيمة خطية، لكن بما يقارب ذلك. من البديهي أن هذا المثال بسيط وأنه مستخدم للتوضيح فقط. يستخدم المبرمجون مكتبات تسمح لهم بتحديد وقت التنفيذ، ووإنشاء والتواصل بين العمليات في نظام موزع (مثل PVM و MPI).

1.2 كيف يمكننا أن نبرمج بحيث نستفيد من الحوسبة المتزامنة؟

هناك طرق عديدة للتعبير عن التكرار في برنامج. أكثرها شيوعاً ما يلي:

١) استخدام خيوط (أو عمليات).

٢) استخدام عمليات في معالجات مختلفة تتخاطب عبر رسائل (نظام تمرير الرسائل MPS)

يمكن تنفيذ كلي الأسلوبين على إعدادات عتاد مختلفة (بمشاركة الذاكرة او الرسائل)، لكن أنظمة MPS قد تتضمن مشاكل في التأخير والسرعة في الرسائل على الشبكة، والذي قد يكون عاملاً سلبياً. ولكن مع التقدم في تقنيات الشبكة، فقد

نمت هذه الأنظمة من حيث الشعبية (والعدد). الرسالة بسيطة جداً:

```
send(destination,msg)
recv(origin,msg)
```

إن أكثر واجهات برمجة التطبيقات شيوعاً هذه الأيام هي PVM و MPI، إضافة إلى كونها لا تنحصر في استخدام الخيوط (حتى وإن كانت في مستوى محلي)، أو وجود دخل/خرج ومعالجة متزامنين. ومن ناحية أخرى، في جهاز ذي ذاكرة مشتركة (SHM)، من الممكن فقط استخدام threads، وهناك مشكلة عويصة في قابلية التوسعة، علماً بأن كل المعالجات تستخدم نفس الذاكرة وأن عدد المعالجات في النظام محصور بعرض نطاق الذاكرة.

وباختصار، يمكننا أن نستنتج ما يلي:

١) تزايد الأجهزة متعدة المهام (ومتعددة المستخدمين) المتصلة عبر الشبكة ذات الخدمات الموزعة (NFS و NIS YP).

٢) هناك أنظمة هجينة بأنظمة تشغيل شبكية (NOS) تقدم مجموعة من الخدمات البعيدة والموزعة.

٣) يمكن برمجة التطبيقات الموزعة على مستويات مختلفة:

أ- باستخدام نموذج العميل - الخادم والبرمجة بمستوى منخفض (عبر المقابس sockets).

ب- نفس النموذج، لكن باستخدام واجهة برمجة تطبيقات API عالية المستوى (PVM, MPI).

ت- استخدام نماذج برمجة مختلفة، كالبرمجة الموجهة للكيانات الموزعة (RMI, CORBA, AGENTS, ...).

1.2.1 الجهاز التخيلي المتوازي PVM

إن PVM واجهة برمجة تطبيقات تجعل من الممكن إنشاء - من وجهة نظر التطبيق - عنقود ديناميكي من الحواسيب، والتي تكون جهازاً تخيلياً Virtual Machine – VM. يمكن إنشاء (spawn) المهام ديناميكياً أو التخلص منها (kill)، ويمكن لكل مهمة PVM إرسالة رسالة لغيرها. لا يوجد حدّ لحجم أو عدد الرسائل (اعتماداً على المعايير، رغم أنه قد تكون هناك تجميعات عتاد/نظام تشغيل تضع قيوداً على حجم الرسائل) والنموذج يدعم التسامح مع الخطأ، والتحكم بالموارد، وأن

يكون هجيناً في شبكاته أو مضيفاته.

للنظام (VM) أدوات للتحكم بالموارد (إضافة أو حذف مضيفية من الجهاز الافتراضي)، ونماذج الاتصال المختلفة (blocking send, blocking/nonblocking receive, multicast)، ومجموعات المهام الديناميكية (يمكن إضافة مهمة إلى جموعة أوإزالتها منها ديناميكياً) والتسامح مع الاخطاء (يكتشف VM الخطأ ويمكن أن تتم إعادة ضبطه).

إن معمارية PVM من ناحية مبنية على المراقب pvm3d الموجود في كل جهاز والذي يتواصل عبر UDP، ومن ناحية أخرى تحوي مكتبة PVM (أي libpvm3.a) التي تحوي كل الـ routines لإرسال/استقبال الرسائل، وإنشاء/التخلص من العمليات، والمجموعات، والمزامنة، إلخ. التي ستستخدم التطبيق الموزع.

تتوفر واجهة نصية لـ PVM تجعل من الممكن بدء المراقب، وإنشاء الجهاز التخيلي، وتشغيل التطبيقات، إلخ. يُنصح بتثبيت البرمجية من التوزيعة، حيث أن التصريف يتطلب قدراً من "التفرّغ". لتثبيت PVM على دبيان – على سبيل المثال – يجب علينا أن نضمّن حزمتين (على الأقل): وهما pvm و pvm-dev (الواجهة النصية لـ pvm وأدواتها في الأولى، والمكتبات والترويسات وبقية أدوات التصريف في الثانية). إذا كنا نحتاج فقط للمكتبات لأن التطبيق موجود لدينا بالفعل، فيمكننا تثبيت الحزمة libpvm3 فقط.

لإنشاء تطبيق متوازٍ/ موزع في PVM، يمكننا البدء بإصدار معياري أو بالنظر إلى المعمارية الفيزيائية للمشكلة وتحديد الأجزاء المتزامنة (المستقلة). ستكون هذه الأجزاء مرشحة لتعاد كتابتها كأكواد متوازية. إضافة إلى ذلك، يجب علينا أن نأخذ بعين الاعتبار ما إذا كان من الممكن إبدال الوظائف الجبرية بإصداراتها المتوازية (مثل حزمة الجبر الخطي القابل للتحجيم ScaLapack، المتاحة في دبيان بالأسماء scalapack-pvm | mpich-test | dev, scalapack1-pvm | mpich، حيث يكون بعضها لـ PVM والبعض الآخر لـ MPI). من المفيد أيضاً إيجاد ما إذا كان هناك أي تطبيق متوازٍ مشابه (http://www.epm.ornl.gov/pvm) التي يمكن أن تكون دليلنا لفهم أسلوب إنشاء التطبيق المتوازي.

إن جعل التطبيق متوازياً ليس بالأمر السهل، حيث علينا أن نأخذ بعين الاعتبار قانون Amdahl.

يُنصّ قانون Amdahl على أن مقدار التسريع speedup تحُدُّه تجزئة الكود البرمجي (f) الذي

يمكن جعله متوازياً:

speedup = 1/(1-f)

يشير القانون بأن التطبيق التسلسلي (f=0) يكون التسريع فيه speedup = 1، وعندما يكون الكود باكمله متوازياً f=1، فسيكون التسريع speedup لا نهاية له (infinite)، وبالقيم الممكنة، يعني كون 90% من الأكواد المتوازية تعني تسريعاً قدره 10، لكن عندما تكون f=0,99، فسيكون التسريع يساوي مئة. يمكن تخطي هذه الحدود بالخوارزميات القابلة للتحجيم والنماذج المختلفة للتطبيقات:

١) المعلم – العامل: يُشغِّل المعلم كل العاملين وينسق العمل والدخل/الخرج.

٢) عملية واحدة وبيانات متعددة (SMPD): نفس البرنامج يتم تنفيذه بمجموعات مختلفة من البيانات.

٣) وظيفية: العديد من البرامج التي تقوم بوظائف مختلفة في التطبيق.

تمكننا الواجهة النصية لـ pvm بالأمر add من ضبط الجهاز الافتراضي أثناء إضافة كل العُقد. في كل من هذه العقد، يجب أن يكون هناك مجلد pvm3/bin/LINUX/~ به الملفات الثنائية للتطبيق. يجب أن يتم تحديد المجلد PVM_ROOT بالمكان الذي فيه lib/LINUX/libpvm3.a، و PVM_ARCH=LINUX، والتي يمكن أن يتم وضعها في الملف cshrc/. مثلاً. الصدفة المبدئية للمستخدم (وتكون بشكل عام مستخدم NIS، وإن لم تكن كذلك، فيجب أن يكون نفس المستخدم موجوداً في كل الأجهزة وبنفس كلمة السر) يُفترض أن تكون csh (إذا كنا نستخدم rsh كوسيلة للاتصال البعيد)، ويجب ضبط الملف rhosts./~ لإتاحة الوصول إلى كل عقدة دون كلمة سر. حزمة PVM تتضمن rsh-pvm التي يمكن إيجادها في /usr/lib/pvm3/bin/ حيث يكون rsh موجوداً لـ PVM على وجه الخصوص (ألق نظرة على التوثيق)، حيث أن هناك بعض التوزيعات التي لا تتضمنها لأسباب أمنية. يُنصح بضبط ssh بالمفاتيح العامة للخادم – كما رأينا سابقاً – في ssh/authorized_keys. من مجلد كل مستخدم.

كمثال على برمجة PVM، نعرض برنامجاً من نوع الخادم-العميل، حيث يُنشئ الخادم العقد الأبناء، ويرسل البيانات، وتمرر هذه العقد البيانات لعدد معين من المرات بين العقد الأبناء (تستقبل العقدة الأولى على اليسار قطعة من البيانات،

وتعالجها، ثم ترسلها إلى من تليها على اليمين)، بينما تنتظر العقد الآباء انتهاء كل من العقد الأبناء.

مثال على PVM: الملف master.c

للتصريف في دبيان:

```
gcc -O -I/usr/share/pvm3/include/ -L/usr/share/pvm3/lib/LINUX -o master master.c -lpvm3
```

يجب أن تكون المجلدات المحددة بكل من I- و L- هي المكان الذي توجد فيه pvm3.h و *libpvm على الترتيب.

التنفيذ:

١. شغل المراقب pvmd عبر pvm

٢. نفذ add لإضافة العقد (يمكن تخطي هذا الأمر إذا كانت لدينا عقدة واحدة فقط).

٣. نفذ quit (نغادر pvm، لكنه يستمر بالعمل)

٤. نشغّل master:

```
#include <stdio.h>
#include "pvm3.h"
#define SLAVENAME "/home/nteum/pvm3/client"
main() {
     int mytid, tids[20], n, nproc, numt, i, who, msgtype, loops; float data[10]; int
n_times;
if( pvm_parent() ==PvmNoParent ){
     /*Return if this is the parent or child process */
     loops = 1;
     printf("\n How many children (120)? ");
     scanf("%d", &nproc);
     printf("\n How many child-child communication loops (1 - 5000)? ");
     scanf("%d", &loops); }
     /*Redirects the in/out of the children to the parent */

     pvm_catchout(stdout);
     /*Creates the children */
      numt = pvm_spawn(SLAVENAME, (char**)0, 0, "", nproc, tids);
     /*Starts up a new process, 1st: executable child, 2nd: argv, 3rd :options, 4th
:where, 5th :N.º copies, 6th :matrix of id*/
     printf("Result of Spawn: %d \n", numt);

/*Has it managed?*/
 if( numt &lt; nproc ){
     Printf("Error creating the children. Error code:\n");
     for( i = numt ; i<nproc ; i++ ) {
           printf("Tid %d %d\n",i,tids[i]); }
     for( i = 0 ; i<numt ; i++ ){
           pvm_kill( tids[i] ); } /*Kill the processes with id in tids*/
      pvm_exit();
     exit(); /*Finish*/
 }

/*Start up parent program, initialising the data */
 n = 10;
for( i = 0 ; i<n ; i++ ){
     data[i] = 2.0;}
/*Broadcast with initial data to slaves*/
 pvm_initsend(PvmDataDefault);.
/*Delete the buffer and specify message encoding*/
```

```
 pvm_pkint(&loops, 1, 1);
/*Package data in the buffer, 2nd N.º, 3*:stride*/
pvm_pkint(&nproc, 1, 1);
pvm_pkint(tids, nproc, 1);
pvm_pkint(&n, 1, 1);
pvm_pkfloat(data, n, 1);
pvm_mcast(tids, nproc, 0);

/*Multicast in the buffer to the tids and wait for the result from the children*/
 msgtype = 5;
for( i = 0 ; i < nproc ; i++ ){
     pvm_recv( -1, msgtype );
     /*Receive a message, -1 :of any, 2nd:tag of msg*/
      pvm_upkint( &who, 1, 1 );
     /*Unpackage*/
      printf("Finished %d\n",who);
     }
pvm_exit();
}
```

مثال على PVM: الملف client.c

للتصريف في دبيان:

gcc -O -I/usr/share/pvm3/include/ -L/usr/share/pvm3/lib/LINUX -or client client.c -lpvm3

التنفيذ

هذا ليس ضرورياً، حيث سيشغلها الخادم، لكن يجب أن يكون العميل في /home/nteum/pvm3

```
#include <stdio.h>
#include "pvm3.h" main() {
int mytid;        /*Mi task id*/
int tids[20];     /*Task ids*/
int n, me, i, nproc, master, msgtype, loops; float data[10];
long result[4]; float work();
mytid = pvm_mytid(); msgtype = 0;

pvm_recv( -1, msgtype );
pvm_upkint(&loops, 1, 1);
pvm_upkint(&nproc, 1, 1);
pvm_upkint(tids, nproc, 1);
pvm_upkint(&n, 1, 1);
pvm_upkfloat(data, n, 1);
/*Determines which child it is (0 -- nproc-1) */
for( i = 0; i < nproc ; i++ )
if( mytid == tids[i] ){ me = i; break; }

/*Processes and passes the data between neighbours*/
 work (me, data, tids, nproc, loops);

/*Send the data to the master */
pvm_initsend( PvmDataDefault );
pvm_pkint( &me, 1, 1 );
msgtype = 5;
```

```
master = pvm_parent(); /*Find out who created it */
pvm_send( master, msgtype);
pvm_exit();
}

float work(me, data, tids, nproc, loops)
int me, *tids, nproc; float *data; {
    int i,j, dest; float psum = 0.0, sum = 0.1;
    for (j = 1; j <= loops; j++){
        pvm_initsend( PvmDataDefault );
        pvm_pkfloat( &sum, 1, 1 );
        dest = me + 1;
        if( dest == nproc ) dest = 0;
        pvm_send( tids[dest], 22 );
        i = me - 1;
        if (me == 0 ) i = nproc-1;
        pvm_recv( tids[i], 22 );
        pvm_upkfloat( &psum, 1, 1 );
        }
    }
```

وهناك واجهة رسومية مفيدة جداً تساعد المبرمج (انظر إلى الشكل التالي)، حيث تتصرف كواجهة نصية ومراقب لـ PVM، وتدعى xpvm (ولتثبيتها في دبيان، ثبت الحزمة xpvm)، والتي تجعل من الممكن ضبط الجهاز الافتراضي، وتنفيذ العمليات، وعرض التفاعل بين المهام (الاتصالات)، والحالات، والمعلومات، إلخ.



شكل 1

1.2.2 واجهة تمرير الرسائل:

إن تعريف واجهة برمجة التطبيقات API لواجهة تمرير الرسائل MPI هي نتيجة لعمل منتدى MPI (أو MPIF، حيث تشير F الأخيرة إلى كلمة Forum)، وهو تجمع لأكثر من 40 مؤسسة. لقد تأثرت MPI بمعماريات ولغات وأعمال مختلفة في عالم التوازي، مثل: WRC (من IBM)، و Intel NX/2, Express, nCUBE, Vertex, p4, Parmac، ومساهمات من ZipCode, Chimp, PVM, Chamaleon, PICL. لقد كان الغرض الرئيسي لتصميم MPIF هو تصميم واجهة برمجة تطبيقات لا علاقة لها بأيّ مصرّف أو مكتبة، بحيث يكون من الممكن عمل اتصالات نسخ كفؤة من الذاكرة وإليها، والحوسبة، والاتصال المتزامن، وتنزيل الاتصالات، حيث نعتبر أن هناك اتصالات المعالج الموازي. إضافة إلى ذلك، فهناك بيئات هجينة، بواجهات C و F77 (بما فيها C++ و F90)، حيث الاتصالات يمكن الاعتماد عليها، ويتم معالجة الخطأ

من طرف النظام. وقد احتاجت واجهة برمجة التطبيقات واجهة للبيئات المختلفة (PVM, NX, Express, p4, ...) وتنفيذاً يمكن تطويعه للمنصات المختلفة، بتغييرات ضئيلة لا تتعارض مع نظام التشغيل (thread-safety). لقد تم تصميم واجهة برمجة التطبيقات هذه خصيصاً للمبرمجين الذين استخدموا نموذج تمرير الرسائل MPP في C و F77 للاستفادة من أكثر المزايا التي توفرها أهمية: portability. يمكن تنفيذ MPP على أجهزة متعددة المعالجات، وشبكات WS، وحتى على الأجهزة ذات الذاكرة المشتركة. لا يدعم الإصدار MPI1 (وهو الإصدار الأكثر انتشاراً) الإنشاء الآلي (spawn) للمهام، لكن MPI2 (الذي يتطور بشكل متسارع) يدعم ذلك.

لقد تم تصميم العديد من النواحي للاستفادة من مزايا عتاد الاتصال في الحواسيب المتوازية القابلة للتحجيم Scalable Parallel Computers – SPC، ولقد قبل بالمعيار مصنعو العتاد المتوازي والموزع (SGI, Sun, Cray, HPConvex, IBM, Parsystec ...). هناك إصدارات مجانية freeware (من mpich على سبيل المثال) (وهي متوافقة كلياً مع الإصدارات التجارية التي يصدرها مصنعوا العتاد)، وتتضمن اتصالات نقطة-إلى-نقطة، والعمليات المشتركة، ومجموعات العمليات، وسياقات الاتصالات والهيكليات، ودعم F77 و C، وبيئات التحكم والإدارة والتسجيل profiling. ولكن هناك أيضاً نقاطاً غير مكتملة، مثل: عمليات SHM والتنفيذ عن بعد، وأدوات إنشاء البرامج، والتصحيح، والتحكم بخيوط المعالجة، وإدارة المهام، ووظائف الدخل/الخرج المتزامن (معظم هذه المشكلات الناجمة عن نقص الأدوات قد تم حلها في الإصدار الثاني API MPI2). الوظيفة في MPI1 – في ظل عدم توفر إنشاء آلي للعمليات – بسيطة جداً – علماً بأنه من بين العمليات الكثيرة الناتجة عن وجود المهام –، ومستقلة، وتنفذ نمط أكواد التعليمات المتعددة والبيانات المتعددة (MIMD) بنفسها، وتتواصل عبر استدعاءات MPI. يمكن أن يكون الكود تسلسلياً أو متعدد الخيوط (متزامن)، وتعمل MPI في وضع threadsafe، وبعبارة أخرى، من الممكن استخدام استدعاءات MPI في الخيوط المتزامنة، أثناء إعادة دخول الاستدعاءات.

من المستحسن استخدام [مستودعات] التوزيعة لتثبيت MPI، علماً بأن تصريفها صعب للغاية (بسبب الاعتماديات التي تحتاجها من حزم مختلفة). تتضمن دبيان mpich في الحزمة mpich-bin (أما حزمة mpich فهي أثرية)، وأيضاً mpich-mpd-bin (إصدار لمراقب متعدد الاستخدامات يتضمن دعماً للعمليات القابلة للتحجيم والإدارة والتحكم). تتضمن حزمة mpich-bin هذه المعيار MPI1,2، وأجزاء من MPI2 (مثل parallel in/out). إضافة إلى ذلك، فهذه التوزيعة

تتضمن تنفيذاً آخر لـ MPI تُدعى LAM (حزم *lam والتوثيق في usr/doc/lam-runtime/release.html/). التنفيذان متكافئان – من وجهة نظر MPI – لكن تتم إدارتهما بطريقة مختلفة. يمكن إيجاد كل المعلومات عن Mpich (بعد تثبيت الحزم *mpich) في usr/share/doc/mpi/ (أو في usr/doc/mpi/). إن mpi تحتاج rsh لتعمل في الأجهزة الأخرى، مما يعني أن علينا إدخال مجلد المنزل في الملف rhosts./~ بسطور بالصيغة التالية: host username، للسماح للمستخدم username بالدخول إلى host دون كلمة مرور (نفس الأمر بالنسبة لـ PVM). علينا أن نتذكر بأنه يجب تثبيت الحزمة rshserver في كل الأجهزة، وإذا كان هناك tcpd في etc/inetd.conf/ على rsh.d، فيجب علينا تفعيل المضيفين في etc/hosts.allow/. إضافة إلى ذلك، يجب أن نكون قد ضمنا مجلد المستخدم عبر NFS في كل الأجهزة، ويجب أن يحوي الملف etc/mpich/machines.LINUX/ اسم المضيف لكل الأجهزة المكونة للعنقود (جهاز واحد في كل سطر، ويظهر في الوضع المبدئي localhost). وإضافة إلى هذا، يجب أن تكون الصدفة المبدئية للمستخدم هي csh.

في دبيان، يمكننا تثبيت الحزمة update-cluster لمساعدتنا في الإدارة. إن تثبيت Mpich في دبيان يستخدم ssh بدلاً من rsh لأسباب أمنية، لكن رغم هذا، يوجد رابط من rsh إلى ssh للتوافقية. الفرق الوحيد هو أنه يجب علينا أن نستخدم آليات استيثاق ssh للاتصال دون كلمة سر عبر الملفات ذات العلاقة. إذا لم نفعل ذلك، فسيكون علينا أن ندخل كلمة السر لكل عملية قبل تنفيذها. للسماح بالاتصال بين الأجهزة عبر ssh دون كلمة مرور، يجب علينا اتباع الإجراءات المذكورة في الأقسام التالية. لفحصها، يمكننا تنفيذ ssh localhost، ومن ثم يفترض أن نكون قادرين على الولوج دون كلمة مرور. أبق في بالك أننا إذا ثبتنا Mpich و LAM-MPI، فستسمى mpirun لـ Mpich بالاسم mpirun.mpich، وسيكون mpirun اسم mpirun لـ LAM-MPI. من الضروري أن نتذكر بأن mpirun لـ LAM ستستخدم المراقب lamboot لتشكيل الهيكلية الموزعة للجهاز الافتراضي.

لقد تم تصميم lamboot بحيث يتمكن المستخدمون من تنفيذ البرامج الموزعة دون الحاجة لصلاحيات الجذر (وتجعل من الممكن أيضاً تنفيذ البرامج في الجهاز الافتراضي دون استدعاءات لـ MPI). ولهذا السبب، لتنفيذ mpirun، سيكون علينا تنفيذ ذلك كمستخدم غير الجذر، وتنفيذ lamboot قبل ذلك. يستخدم lamboot ملف إعداد في etc/lam/ للتعريف المبدئي للعُقد (انظر إلى *bhost)؛ من فضلك راجع التوثيق لمزيد من المعلومات (http://www.lam-mpi.org).

لتصريف برامج MMPI، يمكننا استخدام الأمر mpicc (مثل mpicc -o test test.c)، الذي يقبل كل خيارات gcc، رغم أنه يُنصح باستخدام بعض ملفات makefile الموجودة في الملف usr/doc/mpich/examples/ (دون تعديل). من الممكن أيضاً استخدام mpireconfig makefile الذي يستخدم الملف makefile.in كمدخلة لإنشاء الملف makefile والذي يكون تعديله أسهل بكثير. وبعد ذلك يمكننا تنفيذ التالي:

mpirun -np 8 programme أو mpirun.mpich -np 8 programme

حيث np هو عدد العمليات أو المعالجات التي سيعمل فيها البرنامج (وهو 8 في هذه الحالة). يمكننا أن نضع الرقم الذي نريده، حيث سيحاول mpich توزيع العمليات بطريقة متوازنة بين كل الأجهزة في etc/mpich/machine.LINUX/. إذا كانت هناك عمليات أكثر من المعالجات، فسيستخدم mpich خاصية الإبدال swap في جنو/لينكس لمحاكاة التنفيذ المتوازي. في دبيان، وفي المجلد usr/doc/mpich-doc/ (رابط usr/share/doc/mpich-doc/) يمكننا استخدام كل التوثيق بهيئات مختلفة (الأوامر، وAPI MPI، إلخ).

لتصريف MPI ننفذ: mpicc -O -o output output.c

لتنفيذ mpich ننفذ: mpirun.mpich -np N output (حيث N عدد العمليات).

سنرى الآن مثالين (مضمنين في توزيعة Mpich 1,2,X في المجلد usr/doc/mpich/examples/). إن Srtest برنامج بسيط لإنشاء اتصالات بين عمليات نقطة-إلى-نقطة وتحسب cpi قيمة Pi بطريقة موزعة (عبر التكامل).

الاتصال من نقطة إلى قطة: srtest.c

لتصريفها: mpicc -O -o srtest srtest.c

لتشغيل mpich ننفذ: mpirun.mpich -np N srtest (سيسأل عن كلمة المرور بعدد من المرات قدره N إذا لم يكن لدينا وصول مباشر عبر ssh).

لتشغيل LAM ننفذ: mpirun -np N srtest (يجب أن تكون مستخدماً غير root)

```
#include "mpi.h"
#include <stdio.h>
#define BUFLEN 512
int main(int argc, char *argv[]) {
     int myid, numprocs, next, namelen;
     char buffer[BUFLEN], processor_name[MPI_MAX_PROCESSOR_NAME];
     MPI_Status status;
     MPI_Init(&argc,&argv);
           /* Must be placed before other MPI calls, always */
           MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
           MPI_Comm_rank(MPI_COMM_WORLD,&amp;myid);
           /*Integrates the process in a communications group*/
           MPI_Get_processor_name(processor_name,&namelen);
           /*Obtains the name of the processor*/
      fprintf(stderr,"Process %d on %s\n", myid, processor_name);
     strcpy(buffer,"Hello People");
     if (myid ==numprocs1) next = 0;
     else next = myid+1;
if (myid ==0) { /*If it is the initial, send string of buffer*/
           printf("%d Send '%s' \n",myid,buffer);
MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR, next, 99, MPI_COMM_WORLD);
           /*Blocking Send, 1 or :buffer, 2 or :size, 3 or :type, 4or :destination, 5 or :tag, 6
or :context*/
           /*MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR,
           MPI_PROC_NULL, 299,MPI_COMM_WORLD);*/
            printf("%d receiving \n",myid);
/* Blocking Recv, 1 o :buffer, 2 or :size, 3 or :type, 4 or :source, 5 or :tag, 6 or :context, 7
or :status*/
            MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99,
MPI_COMM_WORLD,&status);
           printf("%d received '%s' \n",myid,buffer) }
```

```
else {
        printf("%d receiving \n",myid);
        MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99,
MPI_COMM_WORLD,status);
        /*MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_PROC_NULL,
299,MPI_COMM_WORLD,&status);*/
         printf("%d received '%s' \n",myid,buffer);
        MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR, next, 99, MPI_COMM_WORLD);
        printf("%d sent '%s' \n",myid,buffer);}
    MPI_Barrier(MPI_COMM_WORLD); /*Synchronises all the processes*/
     MPI_Finalize(); /*Frees up the resources and ends*/
    return (0);
}
```

حسابات PI الموزعة: cpi.c

للتصريف: mpicc O أو cpi cpi.c

لتشغيل mpich ننفذ: mpirun.mpich -np -N cpi (سيسأل عن كلمة المرور بعدد من المرات قدره N-1 إذا لم يكن لدينا وصول مباشر عبر ssh).

لتشغيل LAM ننفذ: mpirun -np N cpi (يجب أن تكون مستخدماً غير الجذر).

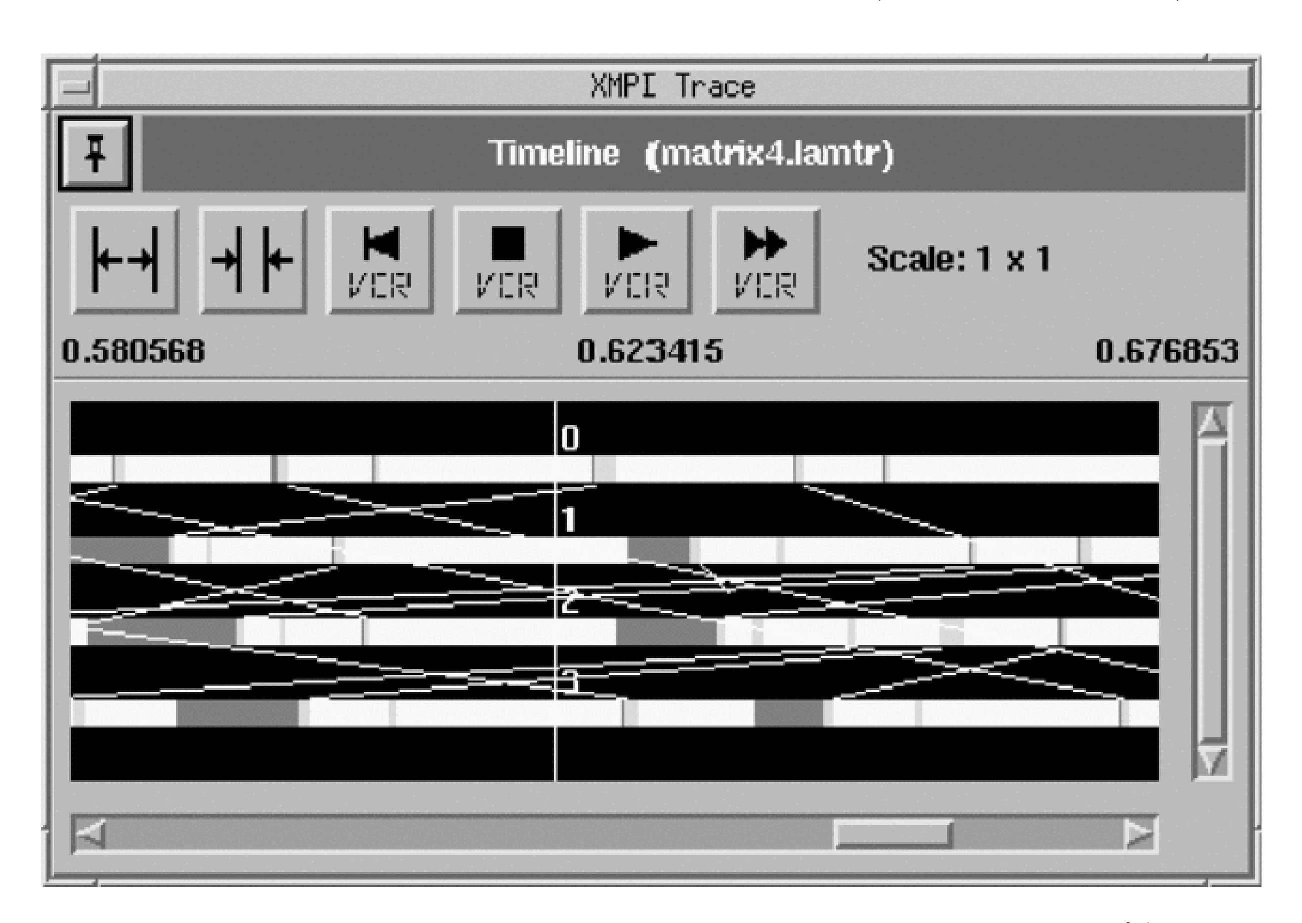
```
 e(); /*Frees up the resources and ends*/
#include "mpi.h"
#include <stdio.h>
#include <math.h>
double f( double );
double f( double a) { return (4.0 / (1.0 + a*a)); }
int main( int argc, char *argv[] ) {
    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;

    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&amp;numprocs);
        /*Indicates the number of processes in the group*/
```

```
            MPI_Comm_rank(MPI_COMM_WORLD,&myid);
           /*Id of the process*/
            MPI_Get_processor_name(processor_name,&namelen);
           /*Name of the process*/
fprintf(stderr,"Process %d on %s\n", myid, processor_name);
n = 0;
while (!done) {
     if (myid ==0) {/*If it is the first...*/
            if (n ==0) n = 100; else n = 0;
           startwtime = MPI_Wtime();} /* Time Clock */
            MPI_Bcast(&amp;n, 1, MPI_INT, 0, MPI_COMM_WORLD);
           /*Broadcast to the rest*/
           /*Send from 4th arg. to all the processes of the group.
           All others that are not 0 will copy the buffer from 4 or arg -process 0-*/
            /*1.º:buffer, 2nd :size, 3rd :type, 5th :group */
            if (n == 0) done = 1; else {
                      h = 1.0 / (double) n;
                      sum = 0.0;
                      for (i = myid + 1; i &lt;= n; i + = numprocs) {
                                x = h * ((double)i - 0.5); sum + = f(x); }
                      mypi = h * sum;
           MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
     MPI_Init(&argc,&argv);
     MPI_Comm_size(MPI_COMM_WORLD,&amp;numprocs);
           /*Indicates the number of processes in the group*/
           MPI_COMM_WORLD);
           /* Combines the elements of the Send Buffer of each process of
           the group using the operation MPI_SUM and returns the result in
           the Recv Buffer. It must be called by all the processes of the group
           using the same arguments*/
           /*1st :sendbuffer, 2nd :recvbuffer, 3rd :size, 4th :typo,
           5th :oper, 6th :root, 7th :context*/
            if (myid == 0){ /*Only the P0 prints the result*/
            printf("Pi is approximately %.16f, the error is %.16f\n", pi, fabs(pi - PI25DT));
           endwtime = MPI_Wtime();
           printf("Execution time = %f\n", endwtime-startwtime); }
           }
     }
```

```
    MPI_Finalize(); /*Free up resources and finish*/
     return 0;
}
```

كما لدى PVM أداة XPVM، فلدى MPI تطبيق مكافئ (أكثر تعقيداً) يُدعى XMPI (الحزمة xmpi). يمكن أيضاً تثبيت المكتبة libxmpi3 التي تتضمن الميفاق XMPI لتحليل برامج MPI رسومياً بتفاصيل أكثر مما توفره xmpi. يعرض الشكل التالي بعض الرسوميات الممكنة في xmpi.



شكل 2: *XMPI*

OpenMosix 2

إن OpenMosix حزمة برمجية تحول مجموعة من الأجهزة المرتبطة عبر شبكة والتي تعمل على جنو/لينكس إلى عنقود. هذا يوازن الحمل تلقائيًّا بين العقد المختلفة في العنقود، ويمكن تجميع العقد، أو ترك العنقود دون مقاطعة الخدمة. الحمل موزع بين العقد، آخذين بعين الاعتبار سرعة الاتصال والمعالج. إن OpenMosix جزء من النواة (عبر ترقيع لنواة لينكس) ويبقي على توافقية كاملة مع جنو/لينكس، و برامج المستخدم، والملفات، والموارد. ومن الصفات الآخرى لـ OpenMosix أنه يتضمن نظام ملفات قوي ومحسن (oMFS) لتطبيقات الحوسبة عالية الأداء HPC. يمكننا في دبيان تثبيت OpenMosix عبر openmosix-dev (المكتبات والترويسات)، و kernel-patch-openmosix (ترقيع OpenMosix)، و openmosix (أدوات إدارية). وكذلك يمكننا تثبيت mosix (انظر للتوثيق للاختلافات - وخاصة فيما يتعلق بالتراخيص - بين Mosix و OpenMosix). لا تتوفر OpenMosix في بعض إصدارات دبيان الحديثة كحزمة (مستقرة) وستحتاج للذهاب إلى http://openmosix.sourceforge.net للحصول على الحزم (أو الموارد) وأدلة التثبيت (http://howto.x-tend.be/openMosix-HOWTO).

يستخدم OpenMosix ملف إعداد يمكن أن يوجد بشكل عام في /etc (راجع التوثيق للإصدارات القديمة من هذا الملف)، ويدعى openmosix.map والتي يمكن أن تكون في كل عقدة. صيغتها بسيطة جداً ويحوي كل سطر ثلاثة حقول: Nodo_ID و عنوان IP (أو اسم المضيف) و Range-size.

ومن الأمثلة على ذلك:

```
1   node1        1
2   node2        1
3   node3        1
4   192،168،1،1  1
5   192،168،1،2  1
```

من الممكن أيضاً استخدام نطاق تتزايد فيه رقم ID وعنوان IP بشكل متوازٍ. علينا التأكد من أن لدينا نفس الإعداد ونفس إصدار OpenMosix في كل عقدة. لتشغيل OpenMosix، علينا أن ننفذ في كل عقدة ما يلي:

```
setpe -w -f /etc/openmosix.map
```

يمكننا أيضاً أن نستخدم النص البرمجي لـ OpenMosix (بنسخها من userspace-tools إلى etc/init.d/) لتشغيلها أثناء الإقلاع.

يسمح نظام الملفات oMFS بالوصول البعيد إلى كل الملفات في العنقود، كما لو كانت مضمومة محلياً. يمكن ضم نظام ملفات العقد الأخرى في mfs/ و – نتيجة لذلك – الملفات في /home/ في العقدة 3 سيصير بالإمكان رؤيتها في كل جهاز في mfs/3/home/.

> يجب أن تكون UID و GID لنظام الملفات في كل عقدة في العنقود متساوية (يمكن استخدام OpenLDAP لعمل هذا).

لضم oMFS يجب علينا تعديل etc/fstab/ بمدخلة مثل: mfs_mnt /mfs mfs dfsa = 1 0 0 ولتفعيلها أو تعطيلها: mfs_mnt /mfs mfs dfsa = 0 0 0.

وبعد ذلك، سيصير بالإمكان رؤية نظام ملفات كل عقدة في مجلد بداخل MFS يحمل اسمه رقم ID للعقدة. بمجرد تثبيته، سيكون من الممكن تنفيذ نص برمجي بسيط جداً مرات عديدة، كالسطر التالي على سبيل المثال (راجع Howto الخاص بـ OpenMosix):

```
awk 'BEGIN {for(i = 0;i<10000;i++)for(j = 0;j<10000;j++);}'
```

ومن ثم يمكننا مراقبة سلوكه عبر mosmom أو openmosixview (مستحسن). OpenMosix له مراقب (omdiscd) يجعل من الممكن ضبط العنقود آلياً بالتخلص من الحاجة لتعديل وضبط etc/openmosix.map/. يستخدم هذا المراقب multicast لتحديد أن العقد الأخرى التي هي أيضاً عقد OpenMosix، مما يعني أنه بمجرد الإقلاع بـ omdiscd، فسينضم هذا المراقب إلى العنقود تلقائياً. ليتم هذا، يجب أن يكون التوجيه المبدئي للشبكة لدينا مضبوطاً بشكل مناسب. بمجرد أن يتم تثبيته (omdiscd)، سيتم إنشاء مجموعة من الرسائل التي تشير إلى حالة العنقود والإعداد. يقدم OpenMosix مجموعة من الأدوات التي يمكن أن يستخدمها المدير لضبط وتضبيط عنقود OpenMosix. يمكن القيام بهذه المهام باستخدام أدوات تعمل في مساحة المستخدم (migrate, mon, mosctl, mosrun) أو عبر واجهة proc/hpc/. من الضروري أن نتذكر بأنه حتى

الإصدار 2,4,16 من OpenMosix كانت الواجهة تدعى proc/mosix/، ولكن منذ الإصدار 2,4,17 صارت تدعى proc/hpc/.

سنقدم الآن ملخصاً بأدوات الإعداد التي يتم تنفيذها في مساحة المستخدم؛ من أجل proc/hpc/ تفقد المراجع:

- migrate [PID] [OpenMosix ID]: يرسل طلب ترحيل إلى خدمة.
- mon: برنامج مراقبة بواجهة نصية يعرض معلومات عن العنقود عبر أعمدة رسم بياني.
- mosctl: هي أداة إعداد OpenMosix. استخدام الخيارات (stay, lstay, block, quiet, mfs, expel, bring, get- tune, getyard, getdecay) يمكننا تحديد ما إذا كان بالإمكان ترحيل العمليات أم لا، واستخدام MFS، والحصول على معلومات عن الحمل، وموازنة الحمل، إلخ.
- mosrun [h | OpenMosix ID | list of OpenMosix IDs] command [argument]: ينفذ أمراً في عقدة معينة.

3 Metacomputers, grid computing

إن متطلبات الحوسبة المطلوبة لتطبيقات معينة كبيرة جداً بحيث تتطلب آلاف الساعات لنتمكن من تنفيذها في بيئات العناقيد. لقد أدت هذه التطبيقات إلى نشوء حواسيب افتراضية على شبكات، أو metacomputers، أو grid computers. لقد جعلت هذه التطبيقات بالإمكان وصل بيئات التنفيذ، والشبكات عالية السرعة، وقواعد البيانات، والأدوات، إلخ.، الموزعة في أقاليم جغرافية مختلفة. هذا يجعل من الممكن تحقيق قدرة على المعالجة لا يمكن مضاهاتها اقتصادياً بأي طريقة أخرى، مع نتائج ممتازة. ومن الأمثلة عل تطبيقاتها بيئات مثل شبكات I-WAY (التي تصل حواسيب خارقة من 17 مكاناً مختلفاً) في أمريكا الشمالية، و DataGrid و CrossGrid في أوروبا، و IrisGrid في إسبانيا. لهذه الحواسيب أو الـ grid computers الكثير من النواحي المشتركة بينها وبين الأنظمة المتوازية والموزعة SPD، لكنها مختلفة أيضاً في نواحٍ هامة عدة. رغم أنها متصلة عبر شبكات، فيمكن أن تكون للشبكات خصائص مختلفة، والخدمات لن تكون مضمونة، ويمكن أن تكون موجودة في نطاقات مختلفة. يجب أن يكون النموذج البرمجي والواجهات البرمجية مختلفين إلى حدّ بعيد (مقارنة بنموذج الأنظمة الموزعة)، ومناسبة للحوسبة عالية الأداء. كما في SPD، تتطلب تطبيقات metacomputing خطة اتصالات لتوفير مستويات الأداء المطلوبة، لكن بأخذ طبيعتها الديناميكية بعين الاعتبار، فهذا يحتاج لأدوات وتقنيات جديدة. وبعبارة أخرى، بينما يمكن تشكيل metacomputing من أساس SPD، فمن الضروري إنشاء أدوات وآليات وتقنيات جديدة لها.

3.1 معماريات مختلفة للحوسبة

إذا كنا نأخذ بعين الاعتبار ناحية القدرة المجمعة فقط، فيمكننا أن نرى بأن هناك حلول عديدة تعتمد على حجم وخصائص المشكلة. بداية، يمكننا أن نفكر باستخدام حاسوب خارق (خادم)، لكننا سنواجه مشاكل مثل عدم قدرتنا على زيادة إمكانياته، والمعدات والصيانة عاليتي التكلفة، حوسبة الذروة (الكثير من موارد الوقت غير المستغلة)، ومشاكل الموثوقية reliability. الحلّ الاقتصادي هو مجموعة من الحواسيب المتصلة ببعضها عبر شبكة عالية الأداء (شبكة إثرنت سريعة LAN أو Myrinet – SAN) التي تشكل عنقوداً من المحطات المتخصصة بالحوسبة المتوازية/الموزعة SPD بمستوى أداء عالٍ جدًّا (قدره 3 – 15 ضعف نسبة الكلفة/الأداء). لكن لهذه الأنظمة عيوب مثل الكلفة العالية للاتصالات، والصيانة، والنموذج البرمجي،

إلخ. ولكنه حل ممتاز للنطاق المتوسط أو حوسبة الوقت العالية HTC. ومن المفاهيم الأخرى المثيرة للاهتمام حوسبة إنترانت، والتي تعني استخدام معدات شبكة محلية (كشبكة من النوع C مثلاً) لتنفيذ وظائف متسلسلة أو متوازية بمساعدة أداة حمل وإدارة، وبعبارة أخرى، هي خطوة تلي العنقود وتسمح باستغلال قدرة الحوسبة في شبكة محلية كبيرة بالمزايا الناتجة عنها، حيث نزيد من كفاءة استخدام الموارد (دارات معالجة منخفضة التكلفة)، ونحسن إمكانية التحجيم، وإدارتها ليست معقدة جداً. لهذا النوع من الحلول، هناك برمجيات مثل Sun Grid Engine من شركة Sun Microsystems، و Condor من جامعة Wisconsin (مجانيان)، و LSF من Platform Computing (تجاري).

إن لخيار حوسبة إنترانت بعض العيوب، كعدم القدرة على إدارة الموارد خارج نطاق الإدارة. بعض الأدوات المذكورة أعلاه (Condor, LSF, SGE) تسمح بالتعاون بين العقد الفرعية المختلفة في النظام، لكن يجب أن تكون فيها كلها نفس معمارية الإدارة، ونفس السياسات الأمنية، ونفس فلسفة إدارة الموارد. ورغم أن هذه خطوة متقدمة فيما يتعلق بالقدرة الحاسوبية بكلفة منخفضة، فهي تدير المعالج فقط، وليس البيانات التي يتم مشاركتها بين العقد الفرعية. إلى جانب ذلك، الموافيق والواجهات مملوكة، وليست مبنية على معيار مفتوح، ولهذا فليس من الممكن استخدام الموارد عندما لا تكون مشغولة بالكامل، ولا يمكننا أيضاً مشاركة الموارد مع المنظمات الأخرى.

إن نمو مجال الحواسيب بين عامي 1986 و 2000 قد تضاعف 500 مرة، وتضاعفت الشبكات 340,000 مرة، لكن المتنبئين كانوا يشيرون إلى أنه بين عامي 2001 و 2010، ستتضاعف الحواسيب 60 مرة، والشبكات 4000 مرة فقط. يشير هذا إلى معيار المعمارية التالية للحوسبة عالية الأداء: الحوسبة الموزعة عبر الإنترنت أو Grid Computing – GC أو metacomputing.

إن Grid Computing تقنية جديدة ناشئة، هدفها مشاركة الموارد عبر الإنترنت بطريقة موحدة، وشفافة، وآمنة، وكفؤة، ويمكن الاعتماد عليها. هذه التقنية مكملة للتقنيات السابقة في كونها تسمح بالتواصل بين الموارد في نطاقات الإدارة المختلفة مع الحفاظ على السياسات الأمنية الداخلية لها، وبرمجيات إدارة الموارد الخاصة بها على إنترانت. ووفقاً لما يقوله إحد آبائها – إيان فوستر Ian Foster – في مقالته "ما هو Grid؟ قائمة تحقق من ثلاث نقاط" (2002)، إن Grid هو نظام:

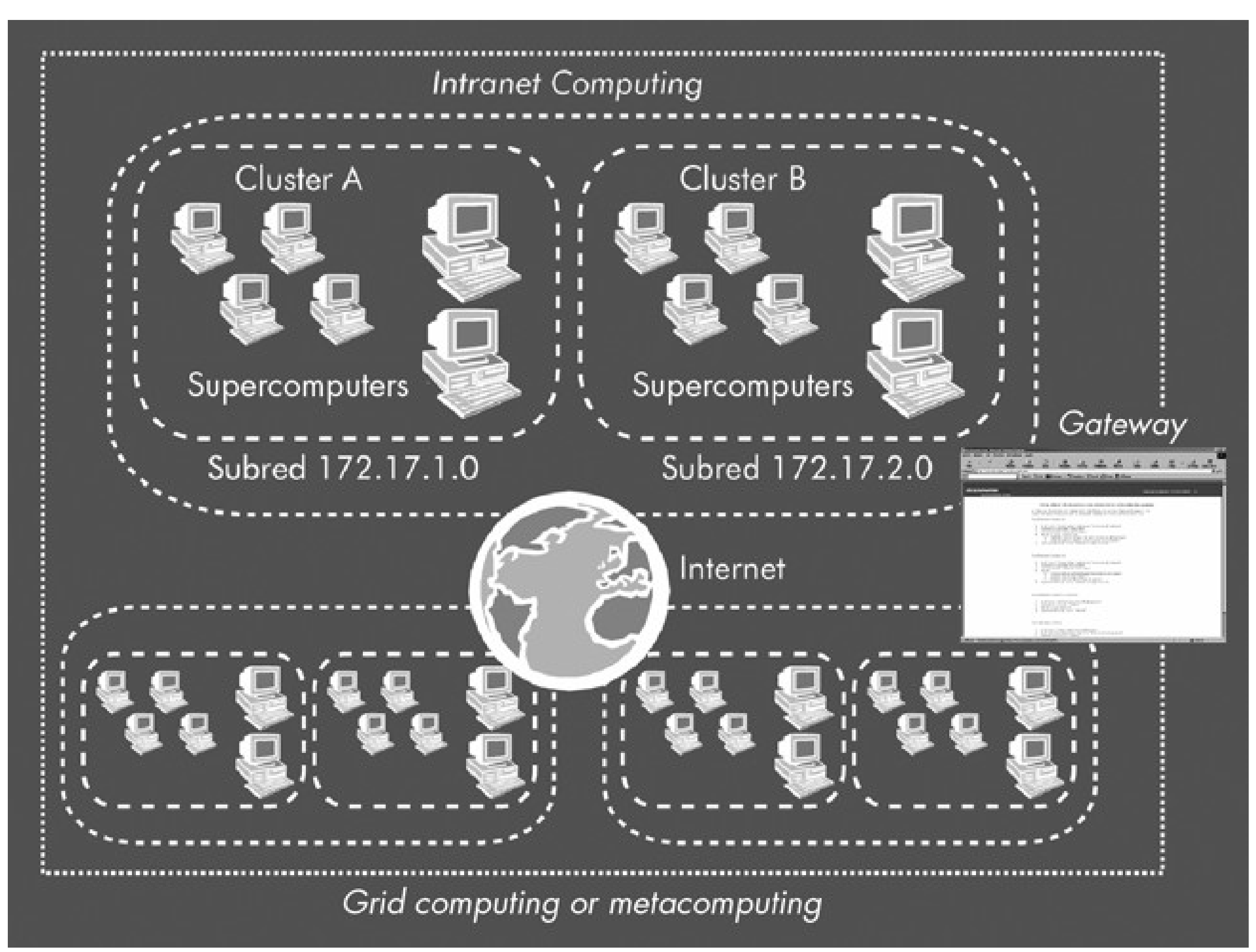
١. ينسق بين الموارد التي لا تتبع تحكماً مركزياً.

٢. باستخدام واجهات وموافيق معيارية مفتوحة عامة الأغراض.

٣. للوصول إلى خصائص غير عادية للخدمة.

من المزايا التي تقدمها هذه التقنية الجديدة، فيمكننا أن نذكر تأجير الموارد، واستغلال الموارد التي لدينا، وقدرات هائلة جداً دون الاضطرار إلى الاستثمار في الموارد والتركيب، والتعاون/المشاركة بين المؤسسات والمنظمات الافتراضية، إلخ.

يقدم الشكل التالي لمحة عن كل هذه المفاهيم.



شكل 3

1.3 جلوبس Globus

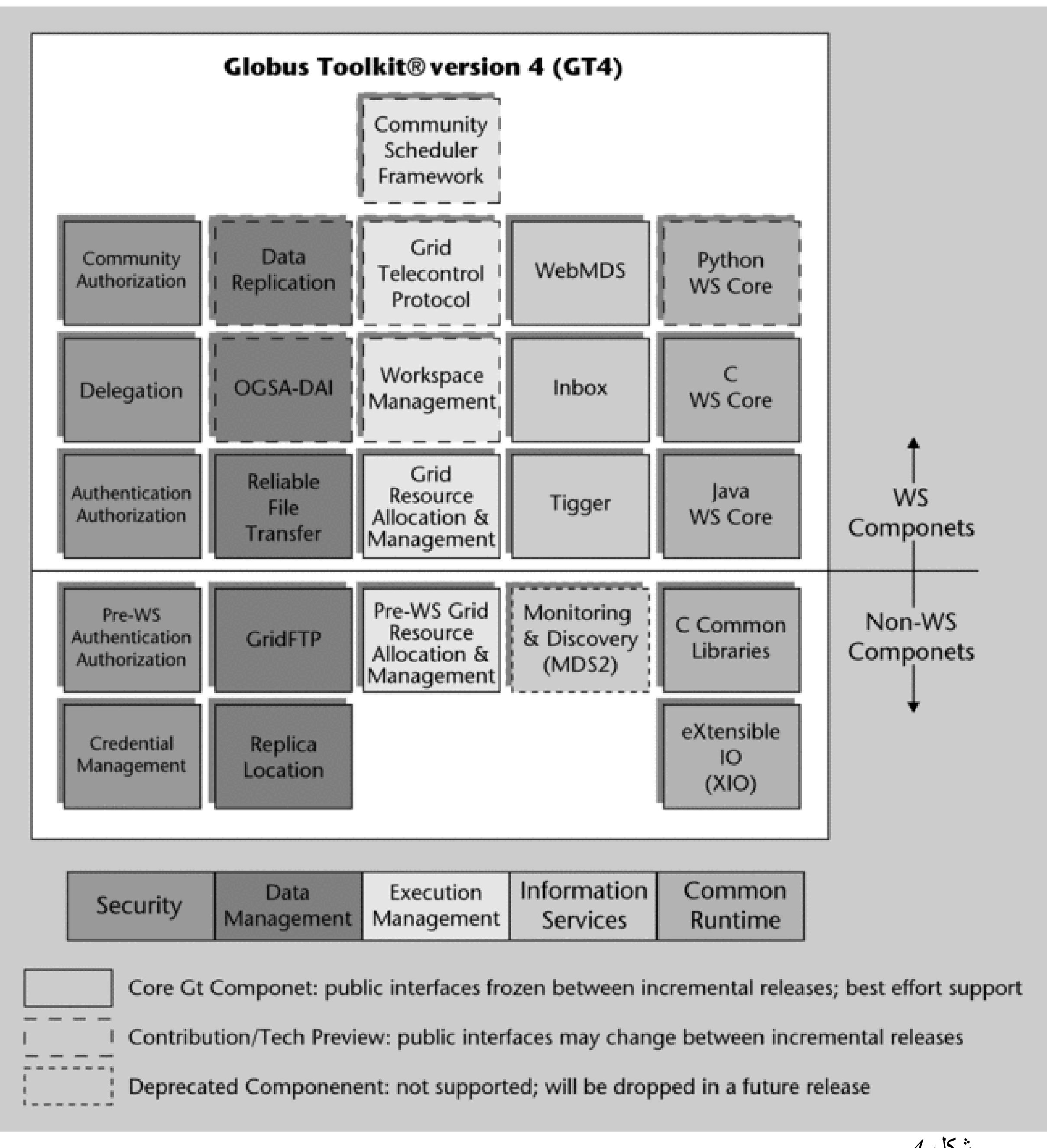
مشروع جلوبس Globus Project هو أحد أكبر رموز هذا المجال، حيث أنه سلف تطوير مجموعة أداة لـ metacomputing و grid computing، وأنه يقدم مزايا هامة في نواح متعلقة بالاتصال، والمعلومات، ومكان وتخطيط الموارد، والاستيثاق، والوصول إلى البيانات. وبعبارة أخرى، فإن جلوبس يجعل من الممكن مشاركة الموارد الموجودة في نطاقات إدارة مختلفة، بسياسات أمنية مختلفة وسياسات مختلفة لإدارة موارد، وهي مكونة من الحزمة البرمجية middleware التي تتضمن المكتبات والخدمات وواجهة برمجة التطبيقات.

تتكون أداة جلوبس (Globus toolkit) من مجموعة من الوحدات بواجهات معرفة جيداً للتفاعل مع الوحدات و/أو الخدمات الأخرى. إن وظائف هذه الوحدات هي كما يلي:

- تحديد وحجز أماكن الموارد؛ يسمح لنا هذا بإخبار التطبيقات عن المتطلبات والموارد التي نحتاجها، آخذين بعين الاعتبار أنه لا يمكن للتطبيق أن يعرف مكان الموارد التي سيعمل عليها.
- الاتصالات؛ يقدم هذا آليات الاتصال الأساسية، والتي تقدم ناحية هامة من النظام، حيث عليها أن تسمح بأساليب مختلفة للتطبيقات لتستخدمها بكفاءة. وتتضمن هذه تمرير الرسائل، واستدعاءات الإجراءات البعيدة RPC، والذاكرة الموزعة المشتركة، وسير البيانات الخطي stream-based dataflow، و multicast.
- تقدم خدمة معلومات الموارد الموحدة Unified Resource Information Service آلية موحدة للحصول على المعلومات في الوقت الحقيقي، كحالة وهيكلية metasystem الذي تعمل عليه التطبيقات.
- واجه الاستيثاق؛ هذه آليات الاستيثاق الأساسية للتحقق من هوية المستخدم والموارد. تنشئ الوحدة الطبقة العلوية التي ستستخدم الخدمات المحلية للوصول إلى بيانات وموارد النظام.
- إنشاء وتنفيذ العمليات؛ ويستخدم هذا لبدء تنفيذ مهمات تم إسناد الموارد إليها، ونقل معاملات التنفيذ والتحكم بها إلى أن يتم التنفيذ.

◆ الوصول إلى البيانات، على هذا أن يقدم وصولاً عالي السرعة إلى البيانات المحفوظة في الملفات. بالنسبة لقواعد البيانات، فتستخدم تقنيات الوصول الموزع أو عبر COBRA، وهي قادرة على تحقيق مستويات أداء عالية عندما يكون لها وصول إلى أجهزة دخل/خرج أو أنظمة ملفات متوازية عبر الشبكة، مثل نظام التخزين عالي الأداء High Performance Storage System – HPSS.

يمكن رؤية البنية الداخلية لجلوبس في الشكل التالي (http://www.globus.org/toolkit/about.html):



شكل 4

1.4 تثبيت وإدارة برمجية جلوبس[1]

إن موقع اتحاد جلوبس 'The Globus Alliance' هو http://www.globus.org. يمكننا أن نجد هنا المصدر البرمجي وكل الوثائق التي يمكن أن نحتاجها لتحويل شبكة إنترانت التي لدينا إلى جزء من grid. بكونها جزءاً من grid، فهذا يعني قبول وتنفيذ سياسات كل المؤسسات والشركات التي تشكل الـ grid. هناك العديد من المبادرات المعتمدة على جلوبس في إسبانيا. ومنها IrisGrid التي يمكننا الانضمام إليها إذا كنا نرغب بالاستفادة من مزايا هذه التقنية. لمزيد من المعلومات، راجع: http://www.rediris.es/irisgrid.

الخطوة الأولى لضبط جلوبس هي الحصول على البرمجية (وهي حالياً Globus Toolkit 5) وتدعى GT5. تطبق هذه البرمجيةُ الخدمات بتجميعة من سي وجافا (وبشكل عام، يمكن تنفيذ مكونات سي في منصات يونكس وجنو/لينكس فقط)، ولهذا فالبرمجية مقسمة إلى الخدمات التي تقدمها. ما يجب تثبيته هو حزم بعينها حسب النظام الذي نرغب بإعداده.

لدليل سريع للتثبيت، شاملاً التنزيل، ومتطلبات النظام، والشهادات يمكن إيجادها في http://www.globus.org/toolkit/docs/5.0/admin/docbook/quickstart.html. وكخلاصة، يجب القيام بالخطوات التالية:

١. قبل البدء: تحقق من البرمجيات وإصداراتها (zlib, j2se, disable gcj, apache, C/C++, tar, make, sed, perl, sudo, postgres, iodbc).

٢. أنشئ مستخدماً، ثم نزّل وصرّف GT5.

٣. شغّل أمن النظام (الشهادات).

٤. شغّل GridFTP.

٥. شغّل حاوي خدمات وب Webservices Container.

---

1 لقد تم تعديل ما يتعلق بـ GT4 من روابط وغيره إلى ما يعادله في الإصدار الأحدث GT5.

٦. اضبط Reliable File Transfer – RFT.

٧. شغّل WS GRAM (إدارة مهام).

٨. شغّل الجهاز الثاني.

٩. شغّل هيكلية خدمة الفهرسة.

١٠.شغل العنقود.

١١.أنشئ Cross-CA Trust.

كما ستلاحظ، فإن تثبيت وإعداد GT5 ليس بالمهمة السهلة، لكنها مبررة إذا كنا نرغب بتضمين عنقود في Grid، أو إذا كنا نرغب بالقيام باختبارات (ننصح بوجود قدر إضافي من الحماس والصبر) للشعور بالقوة الحقيقية لـ GT5. لمعلومات تفصيلية عن تثبيت GT5، من فضلك راجع:

http://www.globus.org/toolkit/docs/5.0/admin/docbook/

الأنشطة

١) ثبّت PVM على عقدة، وشغل البرنامجين master.c و client.c المذكورين كأمثلة، وراقب سلوكهما عبر xpvm.

٢) ثبّت واضبط mpich في عقدة، صرّف ونفّذ البرنامج cpi.c.

٣) ثبّت واضبط LAM-MPI في عقدة، صرف ونفذ البرنامج cpi.c، وراقب سروكه عبر xmpi.

المراجع

مصادر أخرى للمراجع والمعلومات:

[Debc, Ibi, Mou01]

LAM-MPI : http://www.lam-mpi.org/

system-config-cluster (Fedora):
http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.html

OpenMosix: http://openmosix.sourceforge.net/

HowTo OpenMosix: http://howto.x-tend.be/openMosix-HOWTO/

Globus 5: http://www.globus.org/toolkit/5.0/

GT5 Quick Guide:
http://www.globus.org/toolkit/docs/5.0/admin/docbook/quickstart.html

المراجع

المراجع

[Aiv02] **Tigran Aivazian** (2002). "Linux Kernel 2.4 Internals". *The Linux Documentation Project*(guías).

[Ano99] **Anonymous.** *Maximum Linux Security: A Hacker's Guide to Protecting*

[Apa] Apache2 + SSL. <http://www.debian-administration.org/articles/349>.

[Apab] Apache2 + WebDav <http://www.debian-administration.org/articles/285>.

[Apac] Apache2 + Subversion <http://www.debian-administration.org>.

[Ar01] **Jonathan Corbet; Alessandro Rubini**. *Linux Device Drivers 2nd Editon.* O'Reilly, 2001.

[Arc] **Roberto Arcomano.** "Kernel Analysis-HOWTO". *The Linux Documentation Project.*

[Aus] "Australian CERT". <http://www.auscert.org.au/>.

[Bac86] **Maurice J. Bach** (1986). *The Design of the UNIX Operating System.* Prentice Hall.

[Bai03] **Edward C. Bailey** (2003). *RedHat Maximum RPM.* <http://www.redhat.com/docs/books/max-rpm/index.html>.

[Ban] **Tobby Banerjee.** "Linux Installation Strategies HOWTO". *The Linux Documentation Project.*

[Bar] **Slashdot**. *slashdot site.* <http://barrapunto.com>.

[Bas] **Mike G.** "BASH Programming - Introduction HOWTO". *The Linux Documentation Project.*

[Beo] **Beowulf.org**. *Beowulf Web Site.* <http://www.beowulf.org>.

[Bor] **Matthew Borowski** (2000). "FTP". *The Linux Documentation Project.*

[Bro] **Scott Bronson** (2001). " VPN PPP-SSH". *The Linux Documentation Project.*

[Bul] "Bulma Linux User Group". <http://bulmalug.net>.

[Bur02] **Hal Burgiss** (2002). "Security QuickStart HOWTO for Linux". *The Linux Documentation Project.*

[Cac] Monitoring with Cacti. <http://cacti.net/>.

[Cdg] (Environment for portability of GNU/Linux games) <http://www.transgaming.com/>.

[Ced] "Version Management with CVS". <http://www.cvshome.org>.

[Cen] The Comunity ENTerprise Operatyng System <http://www.centos.org>.

[CERa] "CERT site". <http://www.cert.org>.

[CERb] (2003). "CERT vulnerabilities". <http://www.cert.org/nav/index red.htm>.

[Cerc] "Cervisia interface for CVS". <http://cervisia.sourceforge.net>.

[Cis00] (2000). "TCP/IP White Paper". <http://www.cisco.com>.

[Com01] **Douglas Comer** (2001). *TCP/IP Basic principles, protocols and architecture.* Prentice Hall.

[Coo] **Mendel Cooper** (2006). "Advanced bashScripting Guide". *The Linux Documentation Project* (guías).

[CVS] "CVS Home". <http://www.cvshome.org>

[CVSI] Graphic interfaces for CVS <http://www.twobarleycorns.net/tkcvs.html>.

[DBo] **Marco Cesati; Daniel Bovet** (2006). *Understanding the Linux Kernel* (3rd ed.). O'Reilly.

[Deb] "Debian Security Site". <http://www.debian.org/security/>.

[Deb04] (2004). "APT-HOWTO". <http://www.debian.org/doc/manuals/apt-howto/index.en.html>.

[Deba] "Free Software vs Open Software". <http://www.debian.org/intro/free.es.html>.

[Debb] "Debian Distribution". <http://www.debian.org>.

[Dieb] **Hank Dietz** (2004). "Linux Parallel Processing". *The Linux Documentation Project.*

[Dis] "Available Linux distributions". <http://www.distrowatch.com>.

[Dgn] The Dot Gnu Project. <http://www.gnu.org/software/dotgnu/>.

[DNS] Start up a DNS Server. <http://tldp.org/HOWTO/DNS-HOWTO-7.html>.

[Dra] **Joshua Drake** (1999). "Linux Networking". *The Linux Documentation Project.*

[DSL] **Digital Line Subscriber** (2002)*The Linux Documentation Project.*

[Buy] **Kris Buytaert and others** (2002). "The OpenMosix". *The Linux DocumentationProject.*

[Ext] "Extreme Linux Web Site". <http://www.extremelinux.org>.

[Exim] Mail service (MTA). <http://www.exim.org/docs.html>.

[FBI] FBI. "FBI Brigade for cybercrime". <http://www.emergency.com/fbi-nccs.htm>.

[Fed] The Fedora Project. <http://fedoraproject.org>.

[Fen02] **Kevin Fenzi.** "Linux security HOWTO". *The Linux Documentation Project.*

[Fos] (2003). "Globus: A Metacomputing Infrastructure Toolkit". <http://www.globus.org>.

[Fre] "Freshmeat site". <http://freshmeat.org>.

[Fri02] **Aleen Frisch** (2002)*Essential System Administration.* O'Reilly.

[Fry] Monitoring with Frysk. <http://sources.redhat.com/frysk/>.

[FSF] "Free Software Foundation and GNU Project". <http://www.gnu.org>.

[Gar98] **Bdale Garbee** (1998)*TCP/IP Tutorial.* N3EUA Inc.

[Gloa] **Globus. GT4.** *"*Admin Guide Installation" and "Admin Guide Configuration". <http://www.globus.org>.

[Glob] "User's Guide Core Framework Globus Toolkit ", <http://www.globus.org>.

[Gt] **Dirk Allaert Grant Taylor.** "The Linux Printing HOWTO". *The Linux Documentation Project.*

[GT4] Quick Guide. <http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html>.

[Gnu] **Gnupg.org**. *GnuPG Web Site.* <http://www.gnupg.org>.

[Gon] **Guido Gonzato.** "From DOS/Windows to Linux HOWTO". *The Linux Documentation Project.*

[Gor] **Paul Gortmaker** (2003). "The Linux BootPrompt HOWTO". *The Linux Documentation Project.*

[Gre] **Mark Grennan.** "Firewall and Proxy Server HOWTO". *The Linux Documentation Project.*

[Hat01] **Brian Hatch** (2001)*Hacking Linux Exposed.* McGraw-Hill.

[Hat03] (2003). "Firewalls" en Red Hat 9 manual. <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide/ch-fw.html#S1-FIREWALL-IPT>.

[Hatb] (2003). "Red Hat 9 Security Guide". <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide/>.

[Hatc] (2003). "Red Hat Security Site". <http://www.redhat.com/security/>.

[Hatd] **Red Hat** (2003)*Use of GPG signatures in Red Hat.* <http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/ch-gnupg.html>.

[Hen03] **Bryan Henderson.** "Linux Loadable Kernel Module HOWTO". *The Linux Documentation Project.*

[Him01] **Pekka Himanen** (2001). *Hacker ethics and the spirit of the information age* . Destination.

[Hin00] **Martin Hinner.** "Filesystems HOWTO". *The Linux Documentation Project.*

[His] "Linux Hispanic Community". <http://www.hispalinux.es>.

[IET] "Request For Comment Repository developed by the Internet Engineering Task Force (IETF) in the Network Information Center (NIC). ". <http://www.cis.ohio-state.edu/rfc/>.

[Ian] "List of TCP/IP ports". <http://www.iana.org/assignments/port-numbers>.

[IP] Routing with the ip tool. ftp://ftp.inr.ac.ru/ip_routing/>.

[ipw] Firmware for wireless cards IPW2200. <http://ipw2200.sourceforge.net/firmware.php>.

[Ibi] (2003). "Linux Documentation Center". <http://www.ibiblio.org/pub/Linux/docs/HOWTO/>.

[Incb] "vulnerabilities Incidents". <http://isc.incidents.org>.

[Ins] (1998). "Vulnerabilities and exploits". <http://www.insecure.org/sploits.html>.

[Insa] "Insecure.org site". <http://www.insecure.org>.

[Insb] (2003). "Nmap". <http://www.insecure.org/nmap/index.html>.

[Log] LogCheck. <http://logcheck.org/>.

[LWP] LWP: Apache+MySQL+:PHP. <http://www.lawebdelprogramador.com/temas/tema_stablephpapachemysql.php>.

[Joh98] **Michael K. Johnson** (1998). "Linux Information Sheet". *The Linux Documentation Project.*

[Jou] **Linux Journal.** *Linux Journal [Linux Magazine]* . <http://www.linuxjournal.com>.

[Kan] **Ivan Kanis.** "Multiboot with GRUB Mini-HOWTO". *The Linux Documentation Project.*

[Kat] **Jonathan Katz.** "Linux + Windows HOWTO". *The Linux Documentation Project.*

[KD00] **Olaf Kirch; Terry Dawson**. *Linux Network Administrator's Guide.* O'Reilly Associates. And how*e-book*(free) in Free Software Foundation, Inc. , 2000. <http://www.tldp.org/guides.html>.

[Ker02] (2002). "Kernel Hacking Doc Project". <http://www.kernelhacking.org>.

[Kera] "Kernel Newbies". <http://www.kernelnewbies.org>.

[Kerb] "Linux Kernel Archives". <http://www.kernel.org>.

[Kie] **Robert Kiesling** (1997). "The RCS (Revision Control System). ". *The Linux Documentation Project.*

[Knp] Knoppix Distribution. <http://knoppix.org>.

The Linux Documentation Project. [Koe] Kristian Koehntopp. "Linux Partition HOWTO".

[Kuk] **Thorsten Kukuk** (2003). "The Linux NIS(YP)/NYS/NIS+". *The Linux Documentation Project.*

[Lam] "LAM (Local Area Multicomputer). ". <http://www.lam-mpi.org>.

[Law07] **David Lawyer** (2007). "Linux Modem". *The Linux Documentation Project.*

[Lev02] **Bozidar Levi** (2002). *UNIX administration.* CRC Press.

[Lev] "UNIX History". <http://www.levenez.com/unix>.

FHS Standard,[Lin03b] 2003. <http://www.pathname.com/fhs>.

Linux Standards Base project. [Linc] <http://www.linux-foundation.org/en/LSB>.

[Line] **Linuxsecurity.com**. *Linux Security Reference Card.* <http://www.linuxsecurity.com/docs/QuickRefCard.pdf>.

[lkm] **lkml.** *Linux Kernel Mailing List.* <http://www.tux.org/lkml>.

[Llo] **Ignacio Martín Llorente**. *State of Grid Technology and IrisGrid Initiative .* <http://www.rediris.es/irisgrid>.

[Lan] **Nicolai Langfeldt; Jamie Norrish** (2001). "DNS". *The Linux Documentation Project.*

[Log] "Logckeck Web Site". <http://logcheck.org/>.

[LPD] **LPD**. *The Linux Documentation Project.* <http://www.tldp.org>.

[Mag] **Linux Magazine**. *Linux Magazine.* <http://www.linux-mag.com/>.

[Maj96] **Amir Majidimehr** (1996). *Optimizing UNIX for Performance.* Prentice Hall.

[Mal96] **Fred Mallett** (1996). *TCP/IP Tutorial.* FAME Computer Education.

[Mal07] **Luiz Ernesto Pinheiro Malère** (2007). "Ldap". *The Linux DocumentationProject.*

[Miq] **Miquel, S.** "NIS Debian". *On Debian Woody,* /usr/doc/nis/ nis.debian.howto.

[Moin] Moin Moin <http://moinmoin.wikiwikiweb.de/>.

[Moi] Moin Moin + Debian. <http://moinmoin.wikiwikiweb.de/MoinMoinPackages/DebianLinux>.

[Mon] Monit. <http://www.tildeslash.com/monit/>.

[Monb] Monitoring with Munin and monit. <http://www.howtoforge.com/server_monitoring_monit_munin>.

[Monc] Monitoring with SNMP and MRTG. <http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance>.

[Mono] Mono project. <http://www.mono-project.com/Main_Page>.

[Mor03] **Daniel Morill** (2003). *Configuration of Linux systems .* Anaya Multimedia.

[Mou01] **Gerhard Mourani** (2001). *Securing and Optimizing Linux: The Ultimate Solution.* Open Network Architecture, Inc.

[Mun] Munin. <http://munin.projects.linpro.no/>.

[MRTG] MRTG. <http://oss.oetiker.ch/mrtg/>.

[Mur] **Gary Lawrence Murphy**. *Kernel Book Project.* <http://kernelbook.sourceforge.net>.

[Mutt] Mutt mail client. <http://www.mutt.org>.

[Mys] "Reference Manual". <http://www.mysql.com/>.

[MysqlA] <http://www.mysql.com/products/tools/administrator/>.

[Nes] "Nessus". <http://www.nessus.org>.

[Net] **Netfilter.org**.net*filter/IPtables Project.* <www.netfilter.org>.

[Neu] **Christopher Neufeld.** "Setting Up Your New Domain Mini-HOWTO". *The Linux Documentation Project.*

[New] "Newsforge site". <http://newsforge.org>.

[NIS] Setting up a NIS Server. <http://tldp.org/HOWTO/NIS-HOWTO/verification.html>.

[NSAa] "NIST site". <http://csrc.nist.gov/>.

[NSAb] (2003). "Security Enhanced Linux". <http://www.nsa.gov/selinux>.

[Nt3] NTFS-3g Project: NTFS-3G Read/Write Driver. <http://www.ntfs-3g.org/>.

[Oke] **Greg O'Keefe.** "From Power Up To bash Prompt HOWTO". *The Linux Documentation Project.*

[Open] Virtual private network. <http://openvpn.net/howto.html>.

[OpenM] OpenMosix. <http://openmosix.sourceforge.net/>.

[OpenMb] HowTo Openmosix. <http://howto.x-tend.be/openMosix-HOWTO/>.

[OSDa] "Open Source Developement Laboratories". <http://www.osdl.org>.

[OSDb] OSDN. "Open Source Development Network". <http://osdn.com>.

[OSIa] "List of Open Source licenses". <http://www.opensource.org/licenses/index.html>.

[OSIb] (2003). "Open Source Definition". <http://www.opensource.org/docs/definition.php>.

[OSIc] (2003). "Open Source Iniciative". <http://www.opensource.org>.

[Pe 2007) [±ʲ). "Securing Debian Manual". <http://www.debian.org/doc/manuals/securing-debian-howto/>.

[Pga] Client for PostgreSQL. <http://www.pgaccess.org/>.

[Pla] "LSF". <http://www.platform.com>.

[Posa] "PostgreSQL Administrator's Guide". <http://www.postgresql.org/docs/>.

[Per] Performance Monitoring Tools for Linux. <http://www.linuxjournal.com/article.php?sid=2396>.

[Pose] "PostgreSQL Web Site". <http://www.postgresql.org>.

[PPP] **Linux PPP** (2000). "Corwin Williams, Joshua Drake and Robert Hart". *The Linux Documentation Project.*

[Pra03] (2003). "The Wonderful World of Linux 2. 6". <http://www.kniggit.net/wwol26.html>.

[Pri] **Steven Pritchard.** "Linux Hardware HOWTO". *The Linux Documentation Project.*

[Pro] "Grub Manual". <http://www.gnu.org/software/grub/manual/>.

[Proa] "Bastille". <http://bastille-linux.sourceforge.net/>.

[Prob] "MPI". <http://www.mcs.anl.gov:80/mpi/>.

[Proc] "Mpich MPI Freeware". <http://www-unix.mcs.anl.gov/mpi/>.

[Prod] "OpenMosix". <http://openMosix.sourceforge.net>.

[Proe] "PVM Web Site". <http://www.csm.ornl.gov/pvm/>.

[Proc] ProcMail. <http://www.debian-administration.org/articles/242>.

[ProX] Proxy Cache. <http://www.squid-cache.org/>.

[ProT] Transparent Proxy. <http://tldp.org/HOWTO/TransparentProxy-1.html>.

[Prof] ProFTP: FTP file server. <http://www.debian-administration.org/articles/228>.

[PS02] **Ricardo Enr ʲquez Pio Sierra** (2002). *Open Source.* Anaya Multimedia.

[PurF] PureFTP: FTP file server. <http://www.debian-administration.org/articles/383>.

[Qui01] **Ellie Quigley** (2001). *Linux shells by Example.* Prentice Hall.

[Ran] **David Ranch** (2005). "Linux IP Masquerade" and**John Tapsell**. *Masquerading Made Simple.* The Linux Documentation Project.

[Ray98] (1998). "The cathedral and the bazaar". <http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html>.

[Ray02a] **Eric Raymond** (2002). "UNIX and Internet Fundamentals". *The LinuxDocumentation Project.*

[Rayb] **Eric Steven Raymond.** "The Linux Installation HOWTO". *The Linux Documentation Project.*

[Rad] **Jacek Radajewski; Douglas Eadline** (2002). "Beowulf: Installation and Administration". In: Kurt Swendson. *Beowulf HOWTO (tlpd).* <http://www.sci.usq.edu.au/staff/jacek/beowulf>.

[Red] Optimisation of Linux servers. <http://people.redhat.com/alikins/system_tuning.html>.

[Redb] System-config-cluster (FC). <http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.htm>.

[Redh] Red Hat Inc. "Red Hat Distribution". <http://www.redhat.com>.

[Rid] **Daniel Lopez Ridruejo** (2000). "The Linux Networking Overview". *The Linux Documentation Project.*

[Rus] **Rusty Russell.** "Linux IPCHAINS". *The Linux Documentation Project.*

[SM02] **Michael Schwartz and other** (2002). *Multitool Linux - Practical Uses for Open Source Software.* Addison Wesley.

[Sal94] **Peter H. Salus** (1994). "25th anniversary of UNIX" (no. 1, November). *Byte Spain.*

[Sam] Samba Project. <http://samba.org>.

[Sama] Samba HOWTO and Reference Guide (Chapter Domain Control). <http://samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html>.

[Samb] Samba Guide (Chapter Adding Domain member Servers and Clients). <http://samba.org/samba/docs/man/Samba-Guide/unixclients.html>.

[San] "Top20 vulnerabilities". <http://www.sans.org/top20/>.

[Sci] Scientific Linux. <http://www.scientificlinux.org>.

[Sec] **Andr ©s Seco** (2000). "Diald". *The Linux Documentation Project.*

[Sei02] (2002). "Securing Linux, Step by Step". <http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html>.

[Skoa] **Miroslav Skoric.** "LILO mini-HOWTO". *The Linux Documentation Project.*

[Skob] **Miroslav Skoric.** "Linux+WindowsNT mini-HOWTO". *The Linux Documentation Project.*

[Sla] "Slashdot site". <http://slashdot.org>.

[Smb] Wikipedia entry for "Server Message Block". <http://en.wikipedia.org/wiki/Server_Message_Block>.

[Smi02] **Rod Smith** (2002). *Advanced Linux Networking.* Addison Wesley.

[Sno] **Snort.org**. *Snort.* <http://www.snort.org>.

[Sou] "Sourceforge site". <http://sourceforge.org>.

[Squ] Squid proxy server. <http://www.squid-cache.org/>.

[Sta02] (2002). "Discussion by Richard Stallman on relationship between GNU and Linux". <http://www.gnu.org/gnu/linux-and-gnu.html>.

[Stu] **Michael Stutz.** "The Linux Cookbook: Tips and Techniques for Everyday Use". *The Linux Documentation Project*(guías).

[Ste07] Steve French, Linux CIFS Client guide. <http://us1.samba.org/samba/ftp/cifs-cvs/linux-cifs-client-guide.pdf>.

[Stei] (2005). Your Linux Server and Network. Sams.

[Sub] Subversion. <http://subversion.tigris.org>.

[Subb] Control of versions with Subversion. Free Book. <http://svnbook.red-bean.com/index.es.html>.

*[Sun02] **Rahul Sundaram** (2002). "The dosemu HOWTO". *The Linux Documentation Project.*

[Sun] "Sun Grid Engine". <http://www.sun.com/software/gridware/>.

[Tan87] **Andrew Tanenbaum** (1987). *Operating system: Design and Implementation.* Prentice Hall.

[Tan06] **Andrew Tanenbaum; Albert S. Woodhull** (2006). *The Minix Book:*

Operating Systems Design and Implementation (3rd ed.). Prentice Hall.

[Tkc] (2003). "Tkcvs interface for CVS". <http://www.tkcvs.org>. <http://www.twobarleycorns.net/tkcvs.html>.

[Tri] **Tripwire.com.** *Tripwire Web Site.* <http://www.tripwire.com/>.

[Tum02] **Enkh Tumenbayar** (2002). "Linux SMP HOWTO". *The Linux Documentation Project.*

[Ubn] Ubuntu Distribution. <http://www.ubuntu.com>.

[Uni] **Wisconsin University** (2003). *Condor Web Site.* <http://www.cs.wisc.edu/condor>.

[USA] "Division of the US Justice Department for cybercrime". <http://www.usdoj.gov/criminal/cybercrime/>.

[Vah96] **Uresh Vahalia** (1996). *UNIX Internals: The New Frontiers.* Prentice Hall.

[Vas] **Alavoor Vasudevan** (2000). "Modem-Dialup-NT". *The Linux Documentation Project.*

[Vasa] **Alavoor Vasudevan** (2003). "CVS-RCS (Source Code Control System). ". *The Linux Documentation Project.*

[Vasb] **Alavoor Vasudevan.** "The Linux Kernel HOWTO". *The Linux Documentation Project.*

[Wm02] **Matt Welsh and others** (2002). *Running Linux 4th edition.* O'Reilly.

[War] **Ian Ward.** "Debian and Windows Shared Printing mini-HOWTO". *The Linux Documentation Project.*

[Web] **Webmin.** *Tool for administrating Linux systems.* <http://www.webmin.com/>.

[Wil02] **Matthew D. Wilson** (2002). "VPN". *The Linux Documentation Project.*

[Win] Wine Project. <http://www.winehq.com/>.

[Wir] WireShark. <http://www.wireshark.org/download.html>.

[Woo] **David Wood.** "SMB HOWTO". *The Linux Documentation Project.*

[Xin] Xinetd Web Site. <http://www.xinetd.org>.

[Zan] **Renzo Zanelli.** *Win95 + WinNT + Linux multiboot using LILOmini-HOWTO. The Linux Documentation Project.*

GNU Free Documentation License

Version 1.2, November 2002


```
Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```


0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document

straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- **O.** Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
```

```
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.